PARSIMONIOUS ALGORITHMS AND IMPLEMENTATION IN QUANTUM CHEMISTRY

A Dissertation
Presented to
The Academic Faculty

By

Joseph Senan O'Brien

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Computational Science and Engineering
from the College of Sciences

Georgia Institute of Technology

December 2020

PARSIMONIOUS ALGORITHMS AND IMPLEMENTATION IN QUANTUM CHEMISTRY

Thesis committee:

Dr. C. David Sherrill
School of Chemistry and Biochemistry
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Edmond Chow
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Jesse McDaniel
School of Chemistry and Biochemistry
Georgia Institute of Technology

Date approved: December 3, 2020

For those who brought me here.

ACKNOWLEDGMENTS

I would like to thank the members of my thesis committee for their help in preparation of this work especially, my advisor, Prof. C. David Sherrill, whose love of his art I would do well to share. Prof. Sherrill, thank you sincerely for all the opportunities you helped in giving me as well as for your patient advocacy.

I hope that my undergraduate research advisor, Prof. Andrés Cisneros receives my thanks for his patience and wit, as well as for taking me all the way through our project.

My labmates, past and present, as well as Dr. Andy Simmonett and Dr. Monika Kodrycka deserve recognition for their efforts toward my progress. Zach, thank you for your generosity and poise both of which I would do well in trying to match. I wish also to thank my other labmates, Derek for his unique gregariousness, Jeff and Hedieh for their congeniality, Asim and Pavel for their thoughtfulness, Yi and Dom for their ready appreciation of Science, and Prof. Daniel Nascimento his deep appreciation of his craft. Matt, Brandon, and Constance, thank you for your ready welcome to Georgia Tech and your validation for my place in what I did there. Prof. Tu, thank you for your humor and validation. Dr. Simmonett and Dr. Kodrycka, thanks for watching my work and for your suggestions. I would like to thank Dr. Lori Burns for her extensive and ready technical advice without which I would have progressed much more slowly and which was always kindly given. Alice, thank you for your reassurance. Carlos, thank you for your guidance, companionship, understanding, and generosity.

Daniel, thank you for getting me into competitive math. I'd be somewhere else otherwise.

I would like to thank my parents, Tom and Maureen for their encouragement and for their careful attention to my early education and to my moral formation.

My girlfriend, Christa, also deserves thanks for walking through the past years with me as I did. I'm happy you were there.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AO Atomic Orbital

CABS Complementary Auxiliary Basis Set

DF Density Fitting

DFT Density Functional Theory

DirectDFJK Direct Density Fitted Coulomb and Exchange

ERI Electron Repulsion Integral

HF Hartree-Fock

JK Coulomb and Exchange

KS Kohn-Sham

QC Quantum Chemistry

SAPT Symmetry Adapted Perturbation Theory

SCF Self-Consistent Field

SUMMARY

The fundamental crux affecting the performance of quantum chemistry calculations is the need to cover a large number of terms in a way that may entail managing a large amount of data. The costs in time and storage associated with these methods can be mitigated in numerous ways: judiciously exclude insignificant terms, reformulate terms in ways that avoid computing some intermediates, and avoid full formulation of intermediaries. This thesis explores such efforts by examining a direct Density Fitted Coulomb and Exchange (JK) formation algorithm, application of numerous advances in JK construction to Hybrid DFT calculations, reformulation of Coulomb terms to avoid ERI calculations, and parsimonious formation of Coulomb and exchange matrices with different row and column bases under the density fitting approximation.

CHAPTER 1
INTRODUCTION AND BACKGROUND: COULOMB AND EXCHANGE MATRICES

1.1 Hartree Fock Coulomb and Exchange Matrix Formulation

This chapter will serve to remind the reader of the concepts that subtend all the other chapters of the thesis and to introduce newer readers to some of the constructs in Density Fitting (DF)JK operations. The references in the chapter will offer a more complete introduction to these concepts. All parts of this thesis will concern the formation of Fock Matrix components or Fock Matrix component-like objects. The formulation of these objects will be discussed with the necessary context for understanding this thesis, then the Schwarz inequality as it relates to Fock-Matrix formulation will be discussed to explain its extension to the efforts discussed in this thesis.

All the work done in this thesis is concerned with the manipulation of terms $(\mu\nu|P)$[1],[2] which can refer either to 3-index, two electron ERI tensors or the component integrals. Here we will also name some terms for outside Chapter 4, which deals with more exotic theory, as that the Greek letters starting with μ as well as λ and σ will act as indexing variables over the primary atomic orbital basis set for a molecule, the small Roman letter a will index over the molecular orbitals, and the capitol Roman letters starting with P will denote indexing variables running over the fitting basis set for a molecule as in Equation 1.2. The three-index Atomic Orbital (AO) ERI tensors are composed of elements:

$$(\mu\nu|P) = \int dr_1 dr_2 \phi_\mu(r_1)\phi_\nu(r_1)\frac{1}{r_{12}}\hat{\phi}_P(r_2). \tag{1.1}$$

The functions $\phi_\mu \& \phi_\nu$ represent members of a Gaussian basis set, that is a set of contracted Gaussian functions with variances and means that have been parameterized to best form

approximate molecular orbitals for systems containing the various chemical elements. The functions $\hat{\phi}_P$ in Equation 1.1 are the auxiliary basis functions which are also parameterized over atoms but are used in the DF approximation[3]. Auxiliary basis sets (indexed here over P) are different basis sets, generally two to four times the size of the primary/orbital basis sets(indexed here over $\mu\&\nu$.) These basis sets are also used in the construction of the Coulomb Metric $(P|Q)$ which has its elements defined as:

$$(P|Q) = \int \hat{\phi}_P(r_2)\frac{1}{r_{12}}\hat{\phi}_Q(r_2)dr_1dr_2 \tag{1.2}$$

where the circumflex on ϕ communicates the same information as in Equation 1.1. This matrix is diagonalizable, so real numbered powers of it can be taken. It or its p^{th} power is often represented using the notation $[J^p]_{PQ} = (P|Q)^p$ [4]. This thesis only discusses the cases where $p \in \{-1, -\frac{1}{2}\}$.

The integrals in Equation 1.1 can be used to approximate the four-index two electron ERI's:

$$(\mu\nu|\lambda\sigma) = \int dr_1dr_2\phi_\mu(r_1)\phi_\nu(r_1)\frac{1}{r_{12}}\phi_\lambda(r_2)\phi_\sigma(r_2) \tag{1.3}$$

which are used to calculate the Hartree-Fock (HF) Coulomb and Exchange matrices[5] which take the form:

$$J_{\mu\nu} = \sum_{\lambda\sigma}(\mu\nu|\lambda\sigma)D_{\lambda\sigma} = (\mu\nu|\lambda\sigma)D_{\lambda\sigma} \tag{1.4}$$

$$K_{\mu\nu} = \sum_{\lambda\sigma}(\mu\lambda|\nu\sigma)D_{\lambda\sigma} = (\mu\lambda|\nu\sigma)D_{\lambda\sigma}. \tag{1.5}$$

Here we have introduced the Self-Consistent Field (SCF) density matrix $D_{\lambda\sigma}$ which is

calculated from the SCF orbital coefficient matrix $C_{\lambda a}$ as

$$D_{\lambda\sigma} = C_{\lambda a} C_{\sigma a} \tag{1.6}$$

where the orbital coefficients contain the eigenvectors for the largest eigenvalues of the Fock matrix. Note the change to Einstein summation notation that will be used throughout the rest of this thesis.

The HF self-consistent field procedure shares terms with formulations[6] in Kohn-Sham DFT[7] and acts as a precursor to other theoretical treatments of molecules[8]. Using the inverse Coulomb metric Equation 1.2, we can write the Coulomb matrix Equation 1.4 and the exchange matrix Equation 1.5 as:

$$J_{\mu\nu} = (\mu\nu|P)[J^{-1}]_{PQ}(Q|\lambda\sigma)D_{\lambda\sigma} = C_{\lambda a}(\mu\nu|P)[J^{-1}]_{PQ}(Q|\lambda\sigma)C_{\sigma a} \tag{1.7}$$

and

$$K_{\mu\nu} = (\mu\lambda|P)[J^{-1}]_{PQ}(Q|\nu\sigma)D_{\lambda\sigma} = C_{\lambda a}(\mu\lambda|P)[J^{-1}]_{PQ}(Q|\nu\sigma)C_{\sigma a}. \tag{1.8}$$

These equations have a symmetry of terms in that the three-index integrals $(\mu\nu|P)$ need only be determined once for use in these equations. These equations can be reformulated further noting that the Coulomb metric is diagonalizeable, leading us to:

$$J_{\mu\nu} = (\mu\nu|P)[J^{-\frac{1}{2}}]_{PQ}[J^{-\frac{1}{2}}]_{QR}(R|\lambda\sigma)D_{\lambda\sigma} = C_{\lambda a}(\mu\nu|P)[J^{-\frac{1}{2}}]_{PQ}[J^{-\frac{1}{2}}]_{QR}(R|\lambda\sigma)C_{\sigma a} \tag{1.9}$$

and

$$K_{\mu\nu} = (\mu\lambda|P)[J^{-\frac{1}{2}}]_{PQ}[J^{-\frac{1}{2}}]_{QR}(R|\nu\sigma)D_{\lambda\sigma} = C_{\lambda a}(\mu\lambda|P)[J^{-\frac{1}{2}}]_{PQ}[J^{-\frac{1}{2}}]_{QR}(R|\nu\sigma)C_{\sigma a}. \tag{1.10}$$

The elements of these matrices can be calculated using the contracted three index integral terms $B^Q_{\mu\nu} = (\mu\nu|P)[J^{-\frac{1}{2}}]_{PQ}$. Notice that these terms do not depend on either the SCF

orbitals or densities, so they can be saved between iterations. The terms $B^Q_{\mu\nu}$ can be used to calculate the Coulomb and exchange matrices as

$$J_{\mu\nu} = B^Q_{\mu\nu} B^Q_{\lambda\sigma} D_{\lambda\sigma} \tag{1.11}$$

and

$$K_{\mu\nu} = C_{\lambda a} B^Q_{\mu\lambda} B^Q_{\nu\sigma} C_{\sigma a}. \tag{1.12}$$

The formulations in Equation 1.11 and Equation 1.12 are the working equations in Psi4. In equation Equation 1.12, the tensor product: $C_{\lambda a} B^Q_{\mu\lambda} = C_{\lambda a}(\mu\lambda|Q)[J^{-\frac{1}{2}}]_{PQ}$ will be denoted as $X^Q_{\mu a}$. Saving the term $B^Q_{\mu\nu}$ between iterations greatly reduces the time ERI calculation. Some other tensor formulations will be used in calculating these two matrices. The Coulomb Matrix: $J_{\mu\nu}$ can be evaluated in $O(N^2_{bf} N_{aux})$ computational steps with the order of operations:

$$\begin{aligned} J_{\mu\nu} =& B^Q_{\mu\nu} B^Q_{\lambda\sigma} D_{\lambda\sigma} \\ =& B^Q_{\mu\nu} V^Q \end{aligned} \tag{1.13}$$

where we represent the product of $B^Q_{\lambda\sigma} D_{\lambda\sigma}$ as $V^Q = B^Q_{\lambda\sigma} D_{\lambda\sigma}$, a vector over a fitting basis indexed over Q. Equation 1.13 relates the order of operations for Coulomb Matrix construction in the MemDFJK and DiskDFJK in Psi4[9]. The order of operations:

$$\begin{aligned} J_{\mu\nu} =& (\mu\nu|P)[J^{-1}]_{PQ}(Q|\lambda\sigma) D_{\lambda\sigma} \\ =& (\mu\nu|P)[J^{-1}]_{PQ} \Phi^Q \\ =& (\mu\nu|P)\Phi_P \end{aligned} \tag{1.14}$$

can also be useful. Going from the second line to the third line in Equation 1.14 must not be done by multiplication against an inverted Coulomb Metric but must be interpreted as a linear solve using LU decomposition with partial pivoting. Doing otherwise can lead to

conditioning problems so severe that the SCF relying on it will only converge to a limited number of decimal places.

1.1.1 Schwarz Inequality

To hasten the calculation of these terms, it can be advantageous to make use of the Schwarz Inequality to identify sets of ERI's that do not need to be calculated.[10] A statement of the Schwarz Inequality that is particularly useful for DF calculations is

$$(\mu\nu|P) \leq \sqrt{(\mu\nu|\mu\nu)(P|P)}. \tag{1.15}$$

This inequality tells us that a given three-index ERI $(\mu\nu|P)$ is bounded by the integral $(\mu\nu|\mu\nu)$. In practice, Gaussian basis sets are calculated in shells, so if the maximum $(\mu\nu|\mu\nu)\forall\mu,\nu$ in a pair of shells M and N is less than some screening tolerance τ normalized by the maximum $(\lambda\sigma|\lambda\sigma)\forall\lambda,\sigma$ in a primary basis set then all integrals $(\mu\nu|P)\forall\mu,\nu$ in the shell pair M,N can be ignored. Schieber expounds on the savings associated with this screening technique in his thesis[10], and it subtends the work discussed in this document.

A benefit to using Schwarz screening in Density fitting is a reduction in the size of tensor contractions involving AO ERI's. One way to attain these benefits is to observe that for a given AO index μ, only a subset of the AO basis will require its ERI's to be calculated. To reduce memory costs as well as to reduce the time spent in `gemm` operations, only integrals above the scaled threshold should be stored. This leads to a different set of tensor indices for every basis function μ. We will denote a tensor with ERI's stored sparse as $B^{Q}_{\mu\nu^{\mu}}$ to show that the terms traversed by ν depend on each μ.

CHAPTER 2

DIRECTDFJK

This chapter will discuss a subclass for building Fock Matrix components using DF in a way that does not require storing intermediates between iterations in the Restricted HF process which can be extended to non-range separated hybrid DFT calculations on computers with large numbers of processing cores. This chapter will discuss that, the development of the DirectDFJK code by describing first the motivation for the developing the algorithm, second the algorithm decided upon and the merits of the choices made, and third the results for the implementation on various computer architectures with a comparison to the disk-based DF algorithm (Algorithm 1). This algorithm is meant as a schematic, and in practice, steps six through nine are done in blocks based on what can fit in memory.

Algorithm 1 DFHF Memoized Hartree Fock

1: Load Molecule, Bases. Form Core Hamiltonian $H_{\mu\nu}$
2: Populate Tensor $(\mu\nu|P)$ ▷ $O(N_{bf}^2 \times N_{aux})$ `mints` calculations
3: Contract $B_{\mu\nu}^Q = [J^{-\frac{1}{2}}]_{QP}(\mu\nu|P)$ ▷ $O(N_{bf}^2 \times N_{aux}^2)$: `GEMM` operations
4: Form Guess Orbitals, Calculate Guess Energy
5: **while** $\|D_{\mu\nu}^n - D_{\mu\nu}^{n-1}\| > \eta$ **do**
6: **if** Disk **then**
7: Read $B_{\mu\nu}^Q$ ▷ $O(N_{bf}^2 \times N_{aux})$ Serial Disk Reads
8: $J_{\mu\nu} = B_{\mu\nu}^Q B_{\lambda\sigma}^Q D_{\lambda\sigma}$ ▷ $O(N_{bf}^2 \times N_{aux})$: `GEMV` operations
9: $K_{\mu\nu} = C_{\lambda a} B_{\mu\lambda}^Q B_{\nu\sigma}^Q C_{\sigma a}$ ▷ $O(N_{occ} \times N_{bf}^2 \times N_{aux})$: `GEMM` operations
10: Form & Diagonalize Fock Matrix, Update $C_{\mu a}$ & $D_{\mu\nu}$
11: `Return`: Energy and Orbital Occupations

2.1 Background

As many of the timings given in this and later chapters will demonstrate, Fock matrix constructions is the main computational bottleneck in the HF algorithm. In the conventional, non-DF ERI treatment, this is largely due to the need for the computation of $O(N_{bf}^4)$ ERI

terms in Equation 1.4 and Equation 1.5 each of which has a prefactor dependent on the angular momentum number of the atomic orbital basis functions involved and the vectorization of which is still an active area of research. DF can lower the cost of Fock matrix formation through the reformulation of the Coulomb and Exchange terms (Equation 1.11 and Equation 1.12) and by saving the three-index ERI tensors between iterations as in Algorithm 1.

The key weakness of the DFJK algorithm is that it incurs a greater cost in memory than a conventional Direct HF algorithm in which ERI's are calculated as needed. An attempt to circumvent the issues with the direct algorithm for cases in which ERI tensors do not fit in memory involves storing ERI's on either local or network-mounted storage. With this cost, however comes a serial bottleneck in the reading of the ERI tensor $B^Q_{\mu\nu}$ as included in Algorithm 1. Because of the large numbers of processors available to modern machines, this bottleneck can be the rate-limiting step to DF HF calculations as demonstrated on the Psi4[9] quantum chemistry package in Figure 2.1.

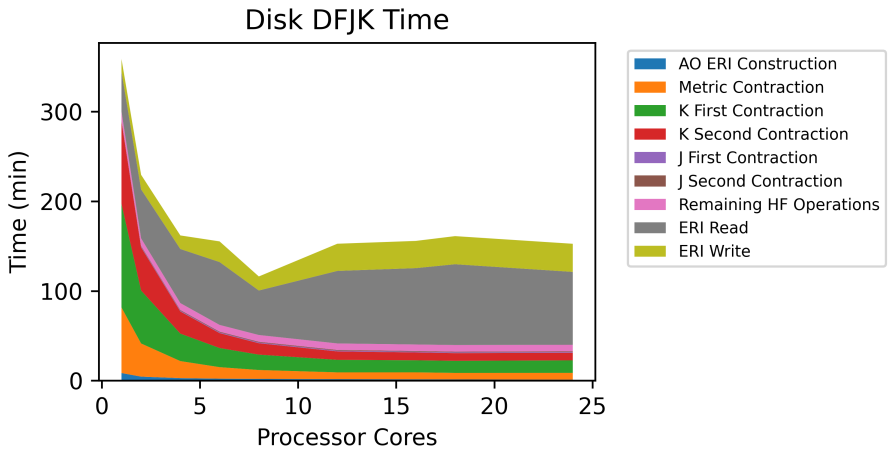

Figure 2.1: Timings of HF calculations on a 40-ane molecule with 1298 aug-cc-pvtz basis functions on an AMD Ryzen Threadripper 3960X 24-Core Processor with 64 GB of DDR4 RAM accessed over 2-channels with scratch configured on a raid array of (3) 2TB 7200 rpm sisk drives.

We can see from Figure 2.1 that for more than five cores, the majority of time taken in our disk-based algorithm is in writing or retrieving information ERI tensors from storage. It might be the case that with this many processing cores, forming the ERI tensors $B^Q_{\mu\nu}$ each iteration of the SCF procedure might be faster than reading them from storage. This work can be viewed as a generalization of the work of Weigend which required that the entire term $(\mu a|P)$ fit in memory.[11]

2.2 Parsimonious Formulation for DirectDFJK

2.2.1 Motivating Reordering of Tensor Contractions

To illustrate the considerations that go into developing an algorithm for a direct construction of ERI tensors, we will show a naïve analog to the Disk based algorithm and the improvements made to it. Timings from pilot implementations of some of the algorithms discussed will be included as well to illustrate the importance that the different choices can have on the cost of the procedures. A discussion of some of the considerations that one must make in choosing a Direct DFJK algorithm is appended.

The first attempt at formulating a Direct DF algorithm for forming Coulomb and exchange matrices is to replace the read step in Algorithm 1 with an ERI calculation step. This will necessarily be followed by a metric contraction step:

$$B^Q_{\mu\nu} = (\mu\nu|P)[J^{-\frac{1}{2}}]_{PQ}. \tag{2.1}$$

However, our problem has already stipulated that this tensor $B^Q_{\mu\nu}$, much less all the intermediates involved in this operation, cannot fit into memory. We must then decide on a piecewise method for calculating the ERI terms that will then be used to calculate the coulomb and exchange matrices.

Blocking Notation

To do so, we will first introduce a notation to help us discuss the blocking procedure. We will block the tensor $\Omega_{\gamma\delta}^{\alpha\beta}$ by defining $[\alpha]_i$ as the i^{th} of N_B blocks over alpha, so that the tensor: $\Omega_{\gamma\delta}^{[\alpha]_i\beta}$ is the i^{th} block of $\Omega_{\gamma\delta}^{\alpha\beta}$ over the index α.

HF Exchange Blocking and Pilot Implementation

We now consider a blocked representation of the HF exchange matrix:

$$K_{\mu\nu} = C_{\lambda a} B_{\mu\lambda}^{[Q]_i} B_{\nu\sigma}^{[Q]_i} C_{\sigma a} \tag{2.2}$$

which is the formulation used in the disk-based algorithm and which we will call the $[Q]_i$ blocking scheme. In the disk algorithm, an ERI block $B_{\mu\nu}^{[Q]_i}$ is read from storage, the orbital coefficient matrices is contracted against it, and the result is accumulated into $K_{\mu\nu}$. This formulation is straightforward, and it has been demonstrated that it works well for a disk-based implementation. One of our pilot implementations substitutes the construction of that ERI block into step 7 of Algorithm 1. Arriving at a working equation consists soley of putting blocks around the index variable Q in Equation 2.1:

$$B_{\mu\nu}^{[Q]_i} = (\mu\nu|P)[J^{-\frac{1}{2}}]_{P[Q]_i} \tag{2.3}$$

which leads us to Algorithm 2.

The runtime cost of this algorithm by component is $O(N_B N_{bf}^2 N_{aux})$ for the ERI construction, $O(N_{bf}^2 N_{aux}^2)$ for the metric contraction, $O(N_B N_{occ} N_{bf}^2 N_{aux})$ for the orbital contraction, and $O(N_{occ} N_{bf}^2 N_{aux})$.

Algorithm 2 and Equation 2.2 present two immediate problems. More than simply computing a set of ERI's each iteration, we will be performing the equivalent to the metric contraction once every iteration. This can be avoided in the disk-based algorithm because

Algorithm 2 Direct DFJK Conventional Blocking and Order of Operations

Given Orbital Occupations $C_{\mu a}$ and densities $D_{\mu\nu}$
for Blocks of Q over i **do**
  for Blocks of P over j **do**
    Populate Tensor $(\mu\nu|[P]_j)$ ▷ $O(N_{bf}^2 \times \frac{N_{aux}}{N_b})$ `mints` calculations
    if i==0 **then**
      Contract $V^{[P]_j} = (\lambda\sigma|[P]_j)D_{\lambda\sigma}$ ▷ $O(N_{bf}^2 \times \frac{N_{aux}}{N_b})$: `GEMV` ops
    if i==1 **then**
      Accumulate $J_{\mu\nu} = (\lambda\sigma|[P]_j)\Phi^{[P]_j}$ ▷ $O(N_{bf}^2 \times \frac{N_{aux}}{N_b})$: `GEMV` ops
    Accumulate $B_{\mu\nu}^{[Q]_i} \mathrel{+}= [J^{-\frac{1}{2}}]_{[Q]_i[P]_j}(\mu\nu|[P]_j)$ ▷ $O(N_{bf}^2 \times \frac{N_{aux}^2}{N_b})$: `GEMM` ops
  Contract $X_{\mu a}^{[Q]_i} = B_{\mu\lambda}^{[Q]_i} C_{\lambda a}$ ▷ $O(N_{occ} \times N_{bf}^2 \times \frac{N_{aux}}{N_b})$: `GEMM` ops
  Accumulate $K_{\mu\nu} \mathrel{+}= X_{\mu a}^{[Q]_i} X_{\nu a}^{[Q]_i}$ ▷ $O(N_{occ} \times N_{bf}^2 \times \frac{N_{aux}}{N_b})$: `GEMM` ops
  if i==0 **then**
    Contract $\Phi^Q = [J^{-1}]_{QP} V^P$

the tensor $B_{\mu\nu}^Q$ can be memoized on disk. Moreover, we must calculate the entire ERI tensor $(\mu\nu|P)$ for every block $B_{\mu\nu}^{[Q]_i}$ that we wish to calculate. We can estimate the increased walltime for such an algorithm as Algorithm 2 by subtracting the reading and writing times from a disk-based Psi4 calculation then by multiplying the metric contraction time by the number of iterations and by multiplying the ERI construction time by the number of read operations. We compare this estimated time to that of the disk-based code in Psi4 in Figure 2.2.

From Figure 2.2 we can see that with some optimizations to the ERI construction handling and to the metric contraction, it may be possible to save time compared to the disk-based algorithm.

2.2.2 Deriving Final Working Equations

To achieve a less costly formulation for the Coulomb and Exchange Matrices, first consider Equation 1.10, restated as:

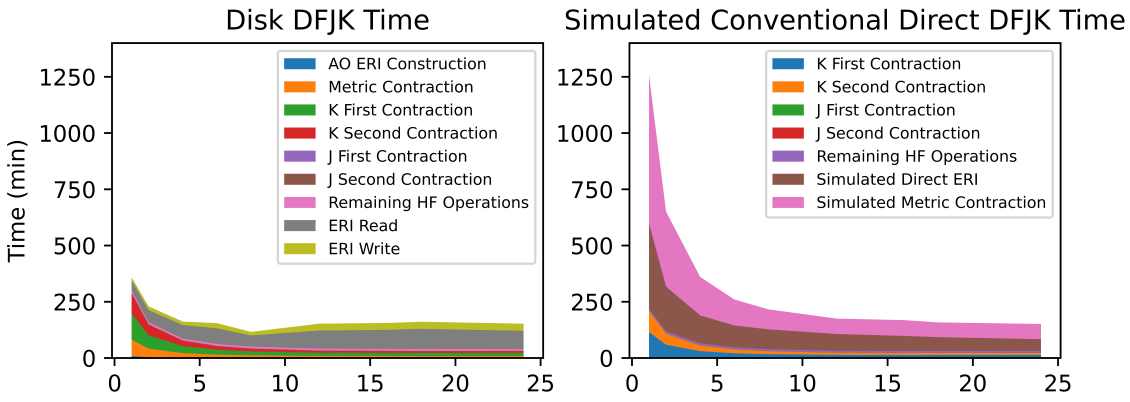

Figure 2.2: Timings of HF calculations on a 40-ane molecule with 1298 aug-cc-pVTZ basis functions on an AMD Ryzen Threadripper 3960X 24-Core Processor with 64 GB of DDR4 RAM accessed over 2-channels (left) alongside the corresponding simulated conventional Direct DFJK calculation (Right). The AO ERI's required 130 GiB Total and 5 disk reads for each of the 9 Fock matrix formations. The penalties from the 5 total recomputations of the ERI Tensors and one full metric contraction per iteration in Algorithm 2 are depicted on the right.

$$K_{\mu\nu} = C_{\lambda a}(\mu\lambda|P)(P|Q)^{-\frac{1}{2}}(Q|R)^{-\frac{1}{2}}(R|\nu\sigma)C_{\nu\sigma}$$

The order of operations in evaluating the right hand side is flexible due to the associative property. In Algorithm 1 and in Algorithm 2, we chose to compute the metric contractions: $(\mu\nu|P)[J^{-\frac{1}{2}}]_{PQ}$ first because the result did not depend on the form of the orbital coefficients $C_{\lambda\sigma}$ and thus could be saved between iterations. However, conducting a full metric contraction requires $O(N_{bf}^2 N_{aux}^2)$ work. This cost can be avoided using the associative property to first evaluate the tensor

$$C_{\lambda a}(\mu\lambda|P)$$

before calculating the tensor product in Equation 2.1. Doing so will not change the cost of the contraction of $C_{\lambda a}$ against an ERI tensor because $B_{\mu\nu}^{Q}$ and $(\mu\lambda|P)$ both have the same size: $N_{bf}^2 \times N_{aux}$ before considering spatial sparsity. The resultant tensor, $(\mu a|P)$ will have

a smaller size: $N_{occ} \times N_{bf} \times N_{aux}$ that leads to a metric contraction of a reduced size:

$$X^Q_{\mu a} = (\mu a|P)[J^{-\frac{1}{2}}]_{PQ} \tag{2.4}$$

which scales as $O(N_{occ}N_{bf}N^2_{aux})$. Notice the use of the notation $X^Q_{\mu a}$ for doubly-contracted ERI tensors, i.e. tensors of ERI's that have been contracted against both the orbital coefficients and the Coulomb metric. Of course, pursuing this reordering eliminates our ability to compute $J_{\mu\nu}$ using the same post-metric contraction ERI tensors as $K_{\mu\nu}$. Fortunately, we can calculate $J_{\mu\nu}$ using the order of operations in Equation 1.14[11]. This will lead to some trade-offs in terms of the number of ERI evaluations, but the trade-off can be considered positive because of the vast reduction in metric contraction time.

Simply making the change to the smaller metric contraction formulation as Equation 2.4 even without making any changes to the blocking scheme in Equation 2.3 leads to a substantial change in the time taken to calculate the complete an SCF calculation as demonstrated in Figure 2.3 which shows a pilot implementation of DirectDFJK algorithms with the contractions in the orders discussed above. That is to say that contracting the raw ERI tensor against the orbital coefficients before contracting against the coulomb metric demonstrably leads to a substantial speed-up (see Figure 2.3).

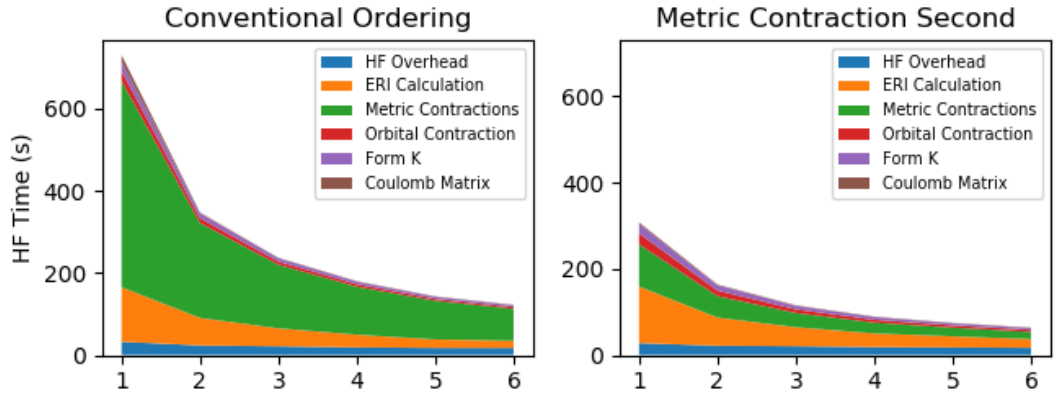

Figure 2.3: Timings of HF Calculations on a glycine dimer with 294 basis functions using an Intel(R) Core(TM) i7-3930K CPU with the contraction in Equation 2.1 on the left and that from Equation 2.4 on the right

The plots in Figure 2.3 help inform the decisions that are made in development of our final algorithm. The cost of metric contraction is reduced by several times by this reordering to being on the same order of magnitude as the ERI calculations. Although the metric contraction is the highest scaling component of the HF time at $O(N_{bf}^2 N_{aux}^2)$, the fact that the ERI calculation portion of Figure 2.3 overtakes the metric contraction portion indicates that algorithmic changes must be made to reduce the $O(N_{bf}^2 N_{aux})$ scaling ERI component of the DirectDFJK based total HF time.

There are two further steps to developing a formulation for the DirectDFJK algorithm and working equations: choosing a tensor formation strategy and choosing a blocking scheme. To motivate the choices made to those ends, we will present a couple figures to motivate the choices made in the blocking scheme then we will discuss spatial sparsity.

2.2.3 Blocking Scheme

We will be expanding the work of [11] to the case where some ERI terms must be recalculated. The central problem of the DirectDFJK algorithm is that core memory cannot hold all the terms in the ERI, so the ERI terms must necessarily be calculated in blocks as in line 4 in Algorithm 2 barring some changes to the surrounding algorithm which we plan to make. Because the exchange matrix represents the majority of time in DF HF calculations, we will construct our blocking scheme with an eye to minimizing work done to compute that matrix. We consider blocking steps over the auxiliary basis set (the conventional $[Q]_i$ blocking scheme) and blocking over the primary basis set (the $[\mu]_i$ blocking scheme.)

To consider the blocking over the auxiliary basis set, restate Equation 2.3:

$$B_{\mu\nu}^{[Q]_i} = (\mu\nu|P)[J^{-\frac{1}{2}}]_{P[Q]_i}$$

for the mathematical form of the ERI block in the $[Q]_i$ scheme. Our working equation for

the exchange matrix accumulates blocks as:

$$K_{\mu\nu} = \sum_i X_{\mu a}^{[Q]_i} X_{\nu a}^{[Q]_i},$$

and observe that forming each block $X_{\mu a}^{[Q]_i} = C_{\lambda a}(\mu\lambda|P)[J^{-\frac{1}{2}}]_{P[Q]_i}$ requires full construction of the ERI tensor $(\mu\lambda|P)$ which scales as $O(N_{bf}^2 N_{aux})$ and the full (K1/small K DGEMM) tensor contraction $(\mu a|P) = C_{\lambda a}(\mu\lambda|P)$ which scales as $O(N_{occ}N_{bf}^2 N_{aux})$ as well as the partial metric contraction $(\mu a|P)[J^{-\frac{1}{2}}]_{P[Q]_i}$. This ordering of the tensor contractions was chosen based on the observations in subsection 2.2.2. Thus, for each SCF iteration, this workflow will require $O(N_B^2 \times N_{occ}\frac{N_{bf}^2}{N_B}N_{aux})$ `GEMM` operations and $O(N_B^2 \times \frac{N_{bf}^2}{N_B}N_{aux})$ ERI evaluations from these repeated operations as well as the metric contraction which requires $O(N_{occ}N_{bf}N_{aux}^2)$ `GEMM` operations in total.

We might consider blocking over the primary basis set indexed by μ in a $[\mu]_i$ scheme rather than the auxiliary basis set indexed by Q. A block $X_{[\mu]_i a}^Q$ is calculated as

$$X_{[\mu]_i a}^Q = C_{\lambda a}([\mu]_i\lambda|P)[J^{-\frac{1}{2}}]_{PQ}$$

which will entail first constructing the eri block $([\mu]_i\lambda|P)$ taking $O(\frac{N_{bf}^2}{N_B}N_{aux})$ ERI evaluationss then conducting the contraction $([\mu]_i a|P) = C_{\lambda a}([\mu]_i\lambda|P)$ taking $O(N_{occ}\frac{N_{bf}}{N_B}N_{aux})$ `GEMM` operations followed by the metric contraction $X_{[\mu]_i a}^Q = ([\mu]_i a|P)[J^{-\frac{1}{2}}]_{PQ}$ which will require $O(N_{occ}\frac{N_{bf}}{N_B}N_{aux}^2)$ `GEMM` operations. These costs per block are reduced relative to the per-block cost for the other blocking procedure.

We seek to write down the per-iteration cost of this blocking scheme. Observe that a block $X_{[\mu]_i a}^Q$ of a doubly-contracted ERI tensor will form a block of the exchange matrix as:

$$K_{[\mu]_i[\nu]_j} = X_{[\mu]_i a}^Q X_{[\nu]_j a}^Q \tag{2.5}$$

which implies that the blocks in this $[\mu]_i$ scheme will require holding two ERI blocks at once. The contraction of these blocks $X_{[\mu]_i a}^Q$ against each other is called the K2/Big K

DGEMM operation in Figure 2.6. We will also have to account that in this case N_B^2 blocks of the exchange matrix, though not necessarily of ERI tensors, are determined in the calculation. As discussed above, decreasing the size of the blocks in the $[\mu]_i$ scheme will, unlike the $[Q]_i$ scheme reduce the cost of constructing each doubly contracted ERI block. Naïvely, we would proceed by simply looping over $i\&j$ to construct the exchange matrix. That scheme would lead to a cost of $O(N_B^2(\frac{N_{bf}^2}{N_B}N_{aux} + N_{occ}\frac{N_{bf}^2}{N_B}N_{aux} + N_{occ}\frac{N_{bf}}{N_B}N_{aux}^2 + N_{occ}\frac{N_{bf}^2}{N_B^2}N_{aux}))$ which would be the same as the cost of the $[Q]_i$ scheme for the ERI calculations and for the (K1/small K DGEMM) operations but would require that the full coulomb metric contraction for each of the blocks in the naïve $[\mu]_i$ scheme. However, we can take advantage of the symmetry of the restricted HF problem and calculate only those blocks $K_{[\mu]_i[\nu]_j}$ of the exchange matrix such that $j \leq i$ then copy these terms above the diagonal. We can further improve that scheme by changing the loop of i over N_B to hold onto the last block of doubly-contracted ERI's from the previous value of i in the loop and by calculating the diagonal blocks $K_{[\mu]_j[\nu]_j}$ alongside the 0^{th} block-wise column of $K_{\mu\nu}$ so that at most one new block of doubly-contracted ERI terms is required for each new block $K_{[\mu]_i[\nu]_j}$. We can also save the last block $X^Q_{[\mu]_i a}$ after each iteration over i to further avoid recomputing ERI's. Formulating all these changes leaves us with Algorithm 3 which makes use of the `AO_Block` operation defined in Algorithm 4 to construct the doubly-contracted ERI tensor $X^Q_{[\mu]_i a}$ and $J_{\mu\nu}$.

We can relate the operations in Algorithm 3 to the order of constructing blocks of the exchange matrix. This is illustrated in Figure 2.4. To calculate off-diagonal blocks of $K_{\mu\nu}$, we must enforce the condition that two blocks $X^Q_{[\mu]_i a}$ be held in memory, so after each block $K_{[\mu]_i[\nu]_j}$ is calculated, a block $X^Q_{[\mu]_i a}$ must be discarded. Algorithm 3 represents a scheme for choosing which blocks to be held at a given time. We first construct $X^Q_{[\mu]_0 a}$ and contract it against itself to construct $K_{[\mu]_0[\nu]_0}$. Then the rest of the diagonal blocks $K_{[\mu]_j[\nu]_j}$ are calculated, and the 0^{th} column of blocks $K_{[\mu]_0[\nu]_j}$ are calculated alongside them. During these steps, the contraction $V^Q += (\mu\nu|Q)D_{\mu\nu}$ is calculated interleaved with the ERI

Algorithm 3 Direct DFJK Parsimonious Symmetry Blocking Algorithm

Require `double` * X_{old}, X_{new} ▷ Double Pointers for doubly contracted terms $X^Q_{[\mu]_i a}$
`Initial_Case` $\leftarrow (N_b < 2?$ `With Contraction` : `Pre Contraction` $)$
`AO_Block(Initial_Case, 0,` X_{old}`)` ▷ Algorithm 4, Stores values in X_{old}
Contract $K_{[\mu]_0[\nu]_0} = X^Q_{[\mu]_{old}a} X^Q_{[\nu]_{old}a}$ ▷ $O(N_{occ} \times \frac{N_{bf}^2}{N_{bf}^2} \times N_{aux})$ GEMM ops
for `i=0` : $N_b - 2$ **do**
  for `j=1` : $N_b - i$ **do**
    if `i==0 && j<`N_b-1 **then**
      `AO_Block(Pre-Contraction, j,` X_{new}`)`
      $K_{[\mu]_j[\nu]_j} = X^Q_{[\mu]_{new}a} X^Q_{[\nu]_{new}a}$ ▷ $O(N_{occ} \times \frac{N_{bf}^2}{N_{bf}^2} \times N_{aux})$ GEMM ops
      $K_{[\mu]_j[\nu]_0} = X^Q_{[\mu]_{new}a} X^Q_{[\nu]_{old}a}$ ▷ $O(N_{occ} \times \frac{N_{bf}^2}{N_{bf}^2} \times N_{aux})$ GEMM ops
    if `i==0 && j==`N_b-1 **then**
      `AO_Block(With Contraction, j,` X_{new}`)`
      $K_{[\mu]_j[\nu]_j} = X^Q_{[\mu]_{new}a} X^Q_{[\nu]_{new}a}$ ▷ $O(N_{occ} \times \frac{N_{bf}^2}{N_{bf}^2} \times N_{aux})$ GEMM ops
      $K_{[\mu]_j[\nu]_0} = X^Q_{[\mu]_{new}a} X^Q_{[\nu]_0 a}$ ▷ $O(N_{occ} \times \frac{N_{bf}^2}{N_{bf}^2} \times N_{aux})$ GEMM ops
    if `i==1` **then**
      `AO_Block( Post-Contraction, j,` X_{new}`)`
      $K_{[\mu]_{row}[\nu]_j} = X^Q_{[\mu]_{old}a} X^Q_{[\nu]_{new}a}$ ▷ $O(N_{occ} \times \frac{N_{bf}^2}{N_{bf}^2} \times N_{aux})$ GEMM ops
    if `i > 1` **then**
      `AO_Block(No-Contraction, j,` X_{new}`)`
      $K_{[\mu]_{row}[\nu]_j} = X^Q_{[\mu]_{old}a} X^Q_{[\nu]_{new}a}$ ▷ $O(N_{occ} \times \frac{N_{bf}^2}{N_{bf}^2} \times N_{aux})$ GEMM ops
  $X_{old} \leftarrow X_{new}$
  $row \leftarrow j$
if $N_B > 1$ **then**
  `AO_Block(Final Coulomb, 1,` X_{new}`)`

Table 2.1: Per-iteration scaling for different DirectDFJK formulations

Components	**Algorithm 2**	$[Q]_i$ Reordered
ERI Construction	$O(N_B N_{bf}^2 N_{aux})$	$O(N_B N_{bf}^2 N_{aux})$
Orbital Contraction	$O(N_B N_{occ} N_{bf}^2 N_{aux})$	$O(N_B N_{occ} N_{bf}^2 N_{aux})$
Metric Contraction	$O(N_{bf}^2 N_{aux}^2)$	$O(N_{occ} N_{bf} N_{aux}^2)$
Final Contraction	$O(N_{occ} N_{bf}^2 N_{aux})$	$O(N_{occ} N_{bf}^2 N_{aux})$

Components	**Algorithm 3**
ERI Construction	$O((N_B + \sum_i^{N_B-2} i) \frac{N_{bf}^2}{N_b} N_{aux})$
Orbital Contraction	$O((N_B + \sum_i^{N_B-2} i) N_{occ} \frac{N_{bf}^2}{N_b} N_{aux})$
Metric Contraction	$O((N_B + \sum_i^{N_B-2} i) N_{occ} \frac{N_{bf}}{N_b} N_{aux}^2)$
Final Contraction	$O(\sum_i^{N_B} i(N_{occ} \frac{N_{bf}^2}{N_B} N_{aux}))$

terms. When j reaches $N_B - 1$ (orange block in Figure 2.4,) the metric contraction for the coulomb terms (middle line in Equation 1.14,) is computed, and the ERI terms are recalculated for the contraction $J_{[\mu]_{N_B-1}\nu} = ([\mu]_{N_B-1}\nu|Q)\Phi^Q$. With the block $X^Q_{[\mu]_{N_B-1-i}a}$ saved, the rest of the blocks of the exchange matrix are calculated and then copied over.

The per-iteration cost for exchange matrix formation using Algorithm 3 is then $O((N_B - 1)N_{occ}\frac{N_{bf}^2}{N_B^2}N_{aux} + (N_B + \sum_i^{N_B-2} i)(\frac{N_{bf}^2}{N_B}N_{aux} + N_{occ}\frac{N_{bf}^2}{N_B}N_{aux} + N_{occ}\frac{N_{bf}}{N_B}N_{aux}^2 + N_{occ}\frac{N_{bf}^2}{N_B^2}N_{aux}))$. Notably, we have reduced the per-iteration blocking prefactor, that is to say, the cost multiplier due to recalculation, to the ERI construction and (K1/small K DGEMM) cost from $\frac{N_B^2}{N_B}$ to $\frac{N_B + \sum_i^{N_B-2} i}{N_B}$ at the cost of changing the blocking prefactor to the metric contraction from $\frac{N_B}{N_B}$ to $\frac{N_B + \sum_i^{N_B-2} i}{N_B}$. We have also reduced the cost of the final exchange contraction $K_{\mu\nu} = X^Q_{\mu a} X^Q_{\mu a}$ by a factor of $\frac{N_B + \sum_i^{N_B-1} i}{N_B^2}$. The parts of the exchange matrix that are calculated at each step of Algorithm 3 are shown in Figure 2.4. Inspection of the algorithm also shows that there is no recalculation of ERI tensors to form the exchange matrix for $N_B < 3$ in the $[\mu]_i$ scheme. To get a sense of the recalculation burden in Figure 2.4, only the red and blue blocks of the exchange matrix require recalculation of ERI terms.

The scaling for all the

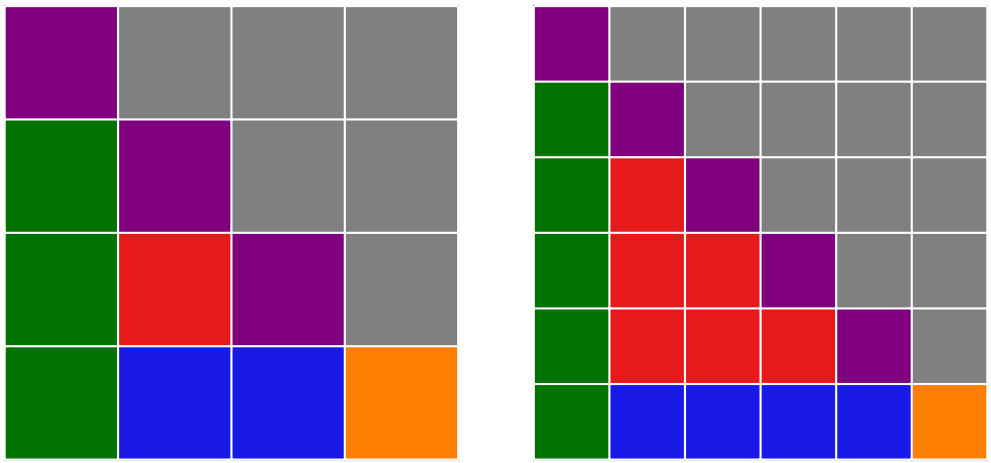

Figure 2.4: Illustration of the regions of the HF exchange matrix constructed in each step of Algorithm 3 with four blocks (left) and six blocks (right.). Colors colors represent the stage of the Fock matrix construction at which each block $K_{[\mu]_i[\nu]_j}$ of the exchange matrix is calculated as well as the kind of work done to calculate the Coulomb matrix and whether ERI tensor blocks are used for only one block according to Algorithm 3. Purple represents work alongside the contraction $V^P = (\mu\nu|P)D_{\mu\nu}$, orange accompanies both $V^P = (\mu\nu|P)D_{\mu\nu}$ and $J_{\mu\nu} = (\mu\nu|Q)\Phi^Q$, and blue, only $J_{\mu\nu} = (\mu\nu|Q)\Phi^Q$. Green Represents blocks $K_{[\mu]_i[\nu]_j}$ reuse ERI blocks.

2.2.4 ERI Tensor Construction.

Our formulation for the DirectDFJK problem does not yet discuss the ERI tensor construction strategy or the calculation of the coulomb matrix. Our strategy will be similar to that of Weigend[11] but will generalize to the case when more than two blocks of doubly contracted ERI's must be calculated, i.e. $X^Q_{[\mu]_i a}$ must be calculated for $i \in \{0, ...N_B\}$ for $N_B \geq 2$. One of the consequences of using the $[\mu]_i$ blocking procedure from Equation 2.5 is that if only the memory footprint of the doubly contracted ERI blocks $X^Q_{[\mu]_i a}$ is considered then our algorithm costs $O(2 \times N_{occ} \frac{N_{bf}}{N_B} N_{aux})$ in memory compared to the $O(\frac{N^2_{bf}}{N_B} N_{aux})$ cost of the DiskDFJK algorithm, a significant reduction in asymptotic scaling for larger basis sets. Keeping the memory cost for the DirectDFJK problem to this amount is imperative for avoiding partitioning the Coulomb and exchange matrix build into more blocks and the accompanying repeated calculations. We can develop a DirectDFJK algorithm without adding any additional $O(n^3_{electron})$ scaling overhead by interleaving the Coulomb matrix

construction, the (K1/small K DGEMM) orbital contraction $(\mu a|P) = C_{\lambda a}(\mu\lambda|P)$, and the shortened metric contraction $(\mu a|Q) = (\mu a|P)[J^{-\frac{1}{2}}]$ with the ERI tensor formation as in Algorithm 4.

Algorithm 4 AO_Block: Integral Construction Handling Cases

Inputs: Coulomb Case, Block Index
for i=First Block Shell **to** Last Block Shell **do** ▷ $O(\frac{N_{bf}}{N_b})$ iterations
  if `Final Coulomb` **then**
    Exit **for** loop
  Compute $([M]_i\nu^\mu|P)$ ▷ $O(N_{bf} \times N_{aux})$ `mints` calculations
  Prune $C_{\nu^\mu a} \longleftarrow C_{\lambda a}$ ▷ $O(N_{bf} \times N_{occ})$ Memory I/O
  Contract $([M]_i a|P) = C_{\nu^\mu a}([M]_i\nu^\mu|P)$ ▷ $O(N_{occ} \times N_{bf} \times N_{aux})$ `GEMM` ops
  Contract $X^Q_{[M]_i a} = [J^{-\frac{1}{2}}]_{QP}([M]_i a|P)$ ▷ $O(N_{occ} \times N_{aux}^2)$ `GEMM` ops
  if `Pre-Contraction` or `With Contraction` **then**
    Prune $D_{M\nu^\mu} \longleftarrow D_{M\nu}$ ▷ $O(N_{bf})$ Memory I/O
    Accumulate $V^P \mathrel{+}= ([M]_i\nu^\mu|P)D_{M\nu^\mu}$ ▷ $O(N_{bf}N_{aux})$ `GEMV` ops
  if `Post-Contraction` **then**
    Contract $J_{M\nu^\mu} = ([M]_i\nu^\mu|P)\Phi^P$ ▷ $O(N_{bf}N_{aux})$ `GEMV` ops
    Unpack $J_{M\nu} \longleftarrow J_{M\nu^\mu}$ ▷ $O(N_{bf})$ Memory I/O
if `With Contraction` **then**
  $\Phi^P = [J^{-1}]_{PQ}V^Q$ ▷ $O(N_{aux}^2)$ `GEMV` ops
  for i=First Block Shell **to** Last Block Shell **do** ▷ $O(\frac{N_{bf}}{N_b})$ iterations
    Compute $([M]_i\nu^\mu|P)$ ▷ $O(N_{bf} \times N_{aux})$ `mints` calculations
    Contract $J_{M\nu^\mu} = ([M]_i\nu^\mu|P)\Phi^P$ ▷ $O(N_{bf}N_{aux})$ `GEMV` ops
    Unpack $J_{M\nu} \longleftarrow J_{M\nu^\mu}$ ▷ $O(N_{bf})$ Memory I/O
if `Final Coulomb` **then**
  for i=First Block Shell **to** Last Block Shell **do** ▷ $O(\frac{N_{bf}}{N_b})$ iterations
    Compute $([M]_i\nu^\mu|P)$ ▷ $O(N_{bf} \times N_{aux})$ `mints` calculations
    Contract $J_{M\nu^\mu} = ([M]_i\nu^\mu|P)\Phi^P$ ▷ $O(N_{bf}N_{aux})$ `GEMV` ops
    Unpack $J_{M\nu} \longleftarrow J_{M\nu^\mu}$ ▷ $O(N_{bf})$ Memory I/O

Here, the capital Greek letter M is used to show that a slice the size of a shell of $(\mu\mu|P)$ is computed in each pass to make use of optimized code for ERI evaluation. The interleaved Coulomb matrix scheme makes use of the formulation in Equation 1.14 to calculate the Coulomb matrix. Its inclusion requires two $O(\frac{N_{bf}^2}{N_B}N_{aux})$ ERI block formations beyond the exchange matrix requirements when $N_B > 1$. That burden is no greater than the ERI recalculation burden for the $[Q]_i$ blocking scheme which is the only alternative within the

constraints of our problem formulation (core memory insufficient for $B^Q_{\mu\nu}$). In all but the worst case scenario, in which the two cases are equal, the $[\mu]_i$ scheme recomputes fewer ERI terms.

The memory layout for arrays storing ERI tensors in the scheme used for DirectDFJK calculations is $\mu Q \nu^{\mu}$, i.e. the address in memory of the term $(\mu Q \nu)$ is offset $\mu \times N_{aux} \times N_{\mu} + Q \times N_{\mu} + \nu$ addresses from the front of the pointer where N_{μ} is the number of primary basis functions ν for which the term $(\mu\nu|Q)$ is kept (see Equation 1.15) . This is done so that Schwarz screening requires no copying outside the trimming of the SCF orbital coefficient matrix $C_{\lambda a} \rightarrow C_{\lambda^{\mu} a}$ which is necessary to make use of dense linear algebra kernels.

2.3 Timings and Comparison to DiskDFJK

Timings of our implementation of the final DirectDFJK algorithm in the Psi4 package and comparisons to the Disk-based code and discussion of both will be included in this section. We will examine the extent to which the algorithmic and formulaic choices we made succeed in mitigating the need to recalculate integrals as well as judge our success in out-performing the disk-based algorithm. This will include a discussion surrounding the extent to which the disk-based algorithm can be replaced by the direct algorithm.

A direct comparison is in Figure 2.5 where we can see a clear advantage to running DirectDFJK on a machine where more than 10 cores are available. The magenta line in Figure 2.6 and in Figure 2.7 is included to compare the performance of the DirectDFJK code to the DiskDFJK code.

Consulting Figure 2.6, we can see some of the benefits of the DirectDFJK algorithm and some of the effects of some of the trade-offs made in optimizing the code. The effects of interleaving the population of ERI tensors $(\mu\nu|P)$ with the formation of exchange matrix intermediates are apparent. We pay a price with the drop in efficiency of the "DDF pQq

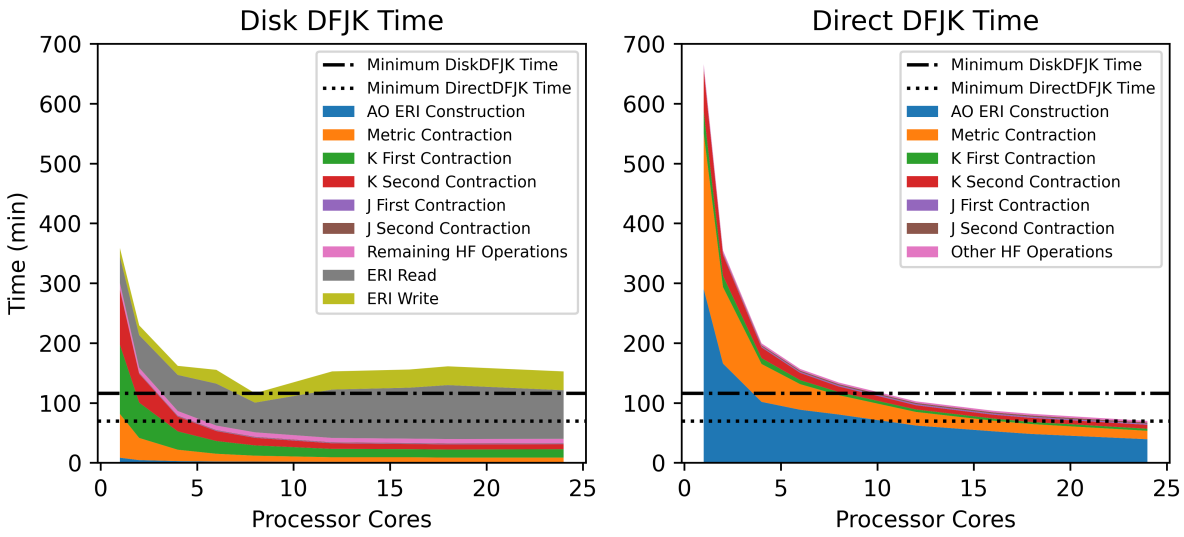

Figure 2.5: Timings of HF Calculations on a 40-ane with 1298 aug-cc-pvtz basis functions using an AMD Ryzen Threadripper 3960X 24-Core CPU

small K DGEMM" (see Figure 2.6 legend) by tasking all processing cores to the `gemm` operation: $(\mu\lambda|P)C_{\lambda a}$ where here μ represents a single basis function rather than all functions indexed by μ. The tensor contraction of the entire blocks $K_{[\mu]_i[\nu]_j} = X^Q_{[\mu]_i a} X^Q_{[\nu]_j a}$ each of which requires several gigabytes of in-core memory also scales poorly after 10 cores, possibly due to the inability to take advantage of memory affinity because ERI terms are initialized by different cores than might act on them during subsequent `gemm` operations. What is true is that each of these operations requires between 200 and 400 seconds over the entire HF calculation with 24 processors tasked which, with 10 HF iterations total for these calculations, amounts to 20 to 40 seconds per transformation which offers a small target for optimization of a calculation that still requires 40 minutes of wall-clock time.

The ERI construction does not parallelize as well as the tensor contractions, but on a machine with two memory channels an algorithm such as this where the data movement scales with the amount of work, this behavior is not surprising. We can also see by comparison to Figure 2.7 that the ERI construction operation: `JK: (A|mn)` scales better than the DirectDFJK code for small numbers of cores. This is also likely due to the interleaving algorithm which leaves DirectDFJK with smaller data over which to spawn threads. This

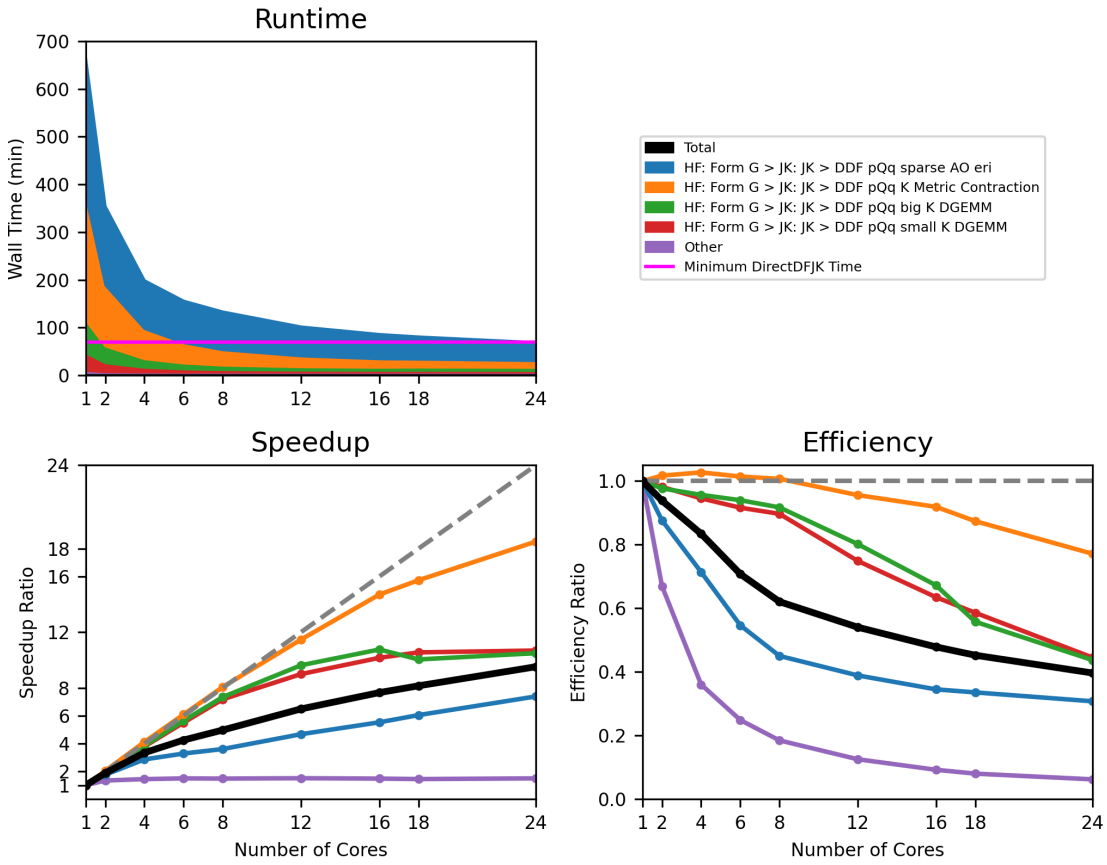

Figure 2.6: Timings of HF Calculations using DirectDFJK on a 40-ane molecule with 1298 aug-cc-pvtz basis functions using an AMD Ryzen Threadripper 3960X 24-Core CPU

.

attribution would agree with the fact that the two codes have the same parallel efficiency at 24 cores where the advantage of larger data is less pronounced.

We also notice that the ERI calculation makes up a larger portion of the time than in Figure 2.3. That is because the pilot implementations kept the entirety of both the terms $(\mu\nu|P)$ and $X^Q_{\mu a}$ in memory for the entirety of the SCF iteration and thus, while being unrealistic for the problem being solved, managed to avoid ERI re-calculation to construct the coulomb matrix. The spatial savings from this holding only $X^Q_{\mu a}$ and from choosing to store intermediates that scaled as $O(N_{occ}N_{bf}N_{aux})$ rather than $O(N^2_{bf}N_{aux})$ in the intermediates kept the number of blocks necessary for computing $K_{\mu\nu}$ down to two. Doing so allows that that all ERI recalculations within a HF iteration were done to compute the

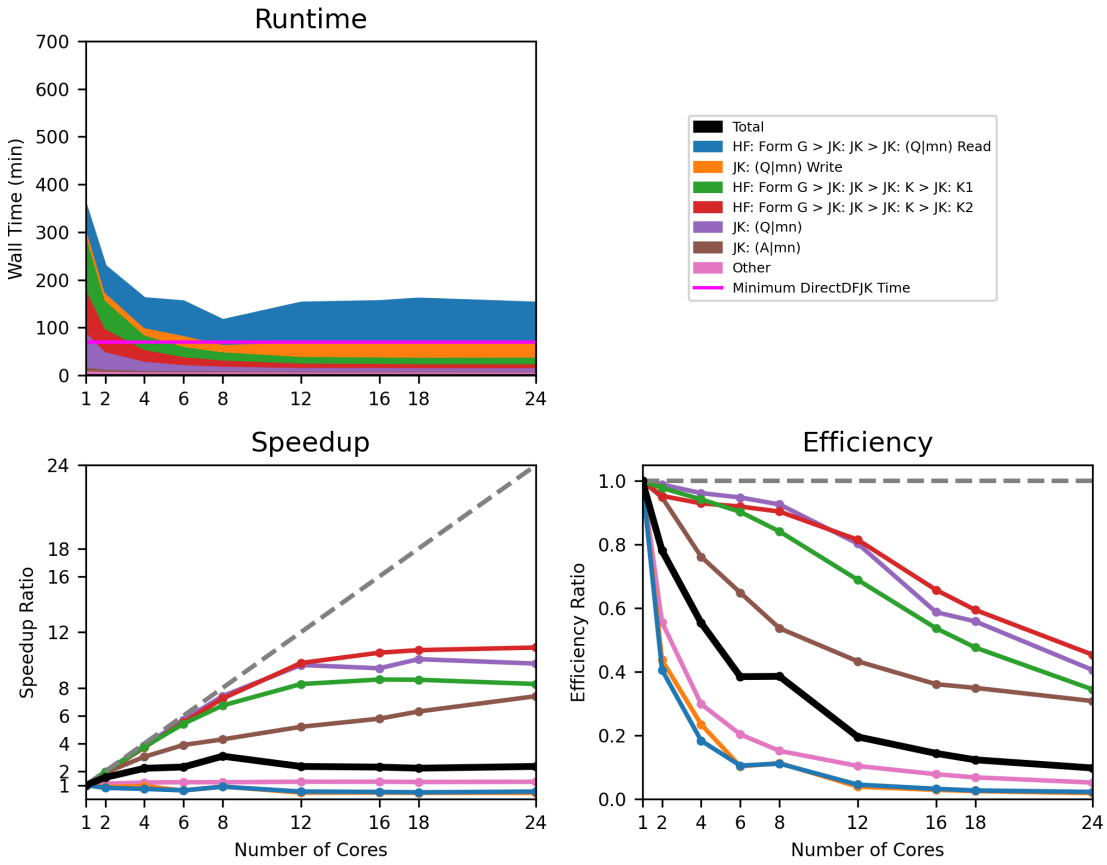

Figure 2.7: Timings of HF Calculations using DiskDFJK on a 40-ane with 1298 aug-cc-pvtz basis functions using an AMD Ryzen Threadripper 3960X 24-Core CPU.

Coulomb matrix whereas the DiskDFJK HF calculations run on the same system required five read operations per iteration. This means that our reformulation calculates ERI terms twice rather than 25 times with each HF iteration, so it becomes difficult to recommend an alternative formulation to the one we have chosen that would jeopardize this benefit.

To demonstrate the effect of requiring more blocks in the DirectDFJK algorithm, Figure 2.8 shows similar timings on the same system in a smaller basis set where less memory is provided to the calculation over several intervals to vary the N_B term in Algorithm 3. The effect is to increase the time cost of the calculation of adding the next block by 25%. Our blocking scheme has, however avoided a quadratic-like shape in the right-hand plot in Figure 2.8.

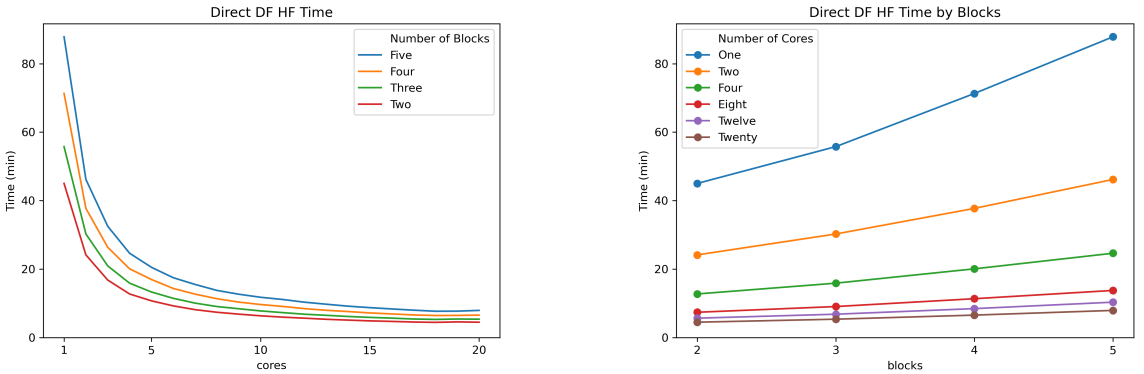

Figure 2.8: Plots showing wall times for DirectDFJK calculations with iso-plots for numbers of ERI blocks (left) and processing of core (right.) Blocks were varied by supplying different amounts of memory to DirectDFJK calculations in Psi4. The calculations are on a 40-ane with 970 basis functions using an Intel Xeon CPU with 20 cores.

Table 2.2: Time taken on silica fragments in the PBE0/aug-cc-pvtz level of theory with 3368 basis functions using DirectDFJK and DiskDFJK on a 20-core Intel Xeon node.

Calculation	5 cores 62GB	10 cores 120GB
DirectDFJK	11.19 Hours	4.01 Hours
DiskDFJK	3.68 Hours	4.25 Hours

Whether or not the DirectDFJK algorithm can replace the DiskDFJK algorithm clearly depends on the amount of memory available and the number of processing cores available. Many of the applications of algorithms that do the work of DirectDFJK involve running large numbers of calculations on shared resources. Conducting such a task allows the user to circumvent in many cases the poor parallelism of many algorithms at large numbers of cores by simply running multiple jobs on the same hardware. DirectDFJK is clearly Figure 2.5 the better choice when a large number of cores must be monopolized by a single calculation. To test the case where many jobs are run on the same resource, a silica fragment was run at the PBE0[6]/aug-cc-pVTZ[12] level of theory. Four five-core jobs were run with 62GB of memory on a node with 254GB of total memory using both the DiskDFJK and the DirectDFJK codes to simulate the performance of these workflows on a shared queue. Similarly, two jobs were run simultanesouly each on 120GB of memory and with 10 cores. The results are in Table 2.2.

We can see the same negative scaling from DiskDFJK as in Figure 2.7 above eight cores, and we can see a significant reduction in the amount of time take in DirectDFJK when more resources are provided to the calculation. It is worth noting that the DirectDFJK calculation calculates ERI's over two blocks in the 10-core job and over four blocks in the five-core job. Disk uses seven reads per SCF iteration in the 62GB job and four reads in the 120GB job.

CHAPTER 3
DFT EXCHANGE IN PSI4

This chapter will discuss three improvements made to the coulomb exchange matrix formation code in Psi4:

1. Extension of the memory layout and sparsity technology in `DFHelper` to range-separated exchange used in DFT calculations.

2. Re-formulation of range-separated exchange to avoid a second contraction of the SCF orbital coefficients.

3. Re-formulation of the HF Coulomb term reduce ERI construction and storage.

3.0.1 Working Exchange Energy Equations

Discussing these changes will necessitate a brief discussion of the formulation of the Exchange terms in Kohn-Sham DFT. A brief summary of the origins of these methods will be presented, followed by a discussion of the working equations for these functionals in the code which has been included in the Psi4 Quantum Chemistry (QC) package. There have been several[6],[13],[14],[15] attempts to arrive at an ideal formulation of the DFT energy of a molecule with a proliferation of different DFT energies and the work of some authors suggesting that no DFT functional can offer accurate energies over all chemical space. To compensate, workers have used the ability of certain formulations to motivate the composition of new functionals to capture a greater portion of chemical space. Among these are the range-separated hybrid DFT functionals which include the HF exchange interaction in their description of the molecule. These functionals are descendants of the work of Savin who used the Ewald summation formulation to motivate the decomposition of the Coulomb

operator in the HF exchange integrals into "long range" and "short range" components[13] as:

$$\frac{1}{r_{12}} = \frac{erf(\omega r_{12})}{r_{12}} + \frac{(1 - erf(\omega r_{12}))}{r_{12}} \tag{3.1}$$

to be used in the calculation of the DFT exchange energy. Building upon Savin's work, Yanai et al.[16] developed what would be called the first range separated hybrid DFT funtional, i.e., a functional that include the HF exchange energy in its energy formulation. In such functionals, the parameter α controls the fraction of the HF exchange energy from that molecule will be included in a given functional's value for that molecule's energy, and the and the β term the amount of range-separated exchange.[16] They decomposed their Coulomb operator further into the equation

$$\frac{1}{r_{12}} = \frac{1 - (\alpha + \beta erf(\omega r_{12}))}{r_{12}} + \frac{\alpha + \beta erf(\omega r_{12})}{r_{12}}. \tag{3.2}$$

Their functional form understands the first term on the right hand side of equation Equation 3.2 as the short range term and the second as the long range term. Notice that in this equation that if we separate the sum on the right hand side further, we arrive at a formulation that includes the regular HF exchange. A way to make use of existing highly optimized QC kernels to calculate range-separated hybrid DFT energies is to decompose these coulomb operators into expressions where all terms involving two-electron ERI's are in terms of normal two-electron ERI's and ERI's that replace $\frac{1}{r_{12}}$ with $\frac{erf(\omega r_{12})}{r_{12}}$. To motivate this discussion, HF exchange energy in the ωB97x functional[17] can be incorporated as follows. The exchange-correlation energy for that functional is written as:

$$E_{xc}^{\omega B97x} = E_x^{LR-HF} + c_x E_x^{SR-HF} + E_x^{SR-B97} + E_c^{B97}. \tag{3.3}$$

The last two terms in this equation are the short-range B97 exchange energy which is not given in terms of two-electron ERI's. The first two terms are called by the authors the

long-range and short-range HF exchange energies and their sum is written as follows:

$$E_x^{LR-HF} + c_x E_x^{SR-HF} = (\mu\lambda|erf(\omega r_{12})|\nu\sigma)D_{\mu\nu}D_{\lambda\sigma} + c_x(\mu\lambda|erfc(\omega r_{12})|\nu\sigma)D_{\mu\nu}D_{\lambda\sigma} \tag{3.4}$$

from which we can rearrange:

$$\begin{aligned} E_x^{LR-HF} + c_x E_x^{SR-HF} &= (\mu\lambda|erf(\omega r_{12})|\nu\sigma)D_{\mu\nu}D_{\lambda\sigma} + c_x(\mu\lambda|erfc(\omega r_{12})|\nu\sigma)D_{\mu\nu}D_{\lambda\sigma} \\ &= (\mu\lambda|erf(\omega r_{12})|\nu\sigma)D_{\mu\nu}D_{\lambda\sigma} + c_x(\mu\lambda|1 - erf(\omega r_{12})|\nu\sigma)D_{\mu\nu}D_{\lambda\sigma} \\ &= (\mu\lambda|(1 - c_x)erf(\omega r_{12})|\nu\sigma)D_{\mu\nu}D_{\lambda\sigma} + c_x(\mu\lambda|\nu\sigma)D_{\mu\nu}D_{\lambda\sigma} \\ &= (1 - c_x)K_{\mu\nu}^{\omega\psi_4}D_{\mu\nu} + c_x K_{\mu\nu}^{\psi_4}D_{\mu\nu}. \end{aligned}$$

Where the $K_{\mu\nu}^{\omega\psi_4}$ term is a matrix that is formed in the same way as the exchange matrix in Psi4 using integrals of the form $(\mu\nu|erf(\omega r_{12})|\lambda\sigma)$ rather than integrals of the form $(\mu\nu|\lambda\sigma)$ as are used to form $K_{\mu\nu}^{\psi_4}$. This equation is specific to the ωB97x functional, and the above discussion is meant to provide physical context to the terms that are being computed in the code discussed in this thesis. For other functionals that may have a coefficient in front of the Slater exchange term and that might not have the same physical attribution to their terms as ωB97x, we will state the working equation for calculating HF exchange terms for DFT as:

$$E_x = \alpha^{\psi 4} \times K_{\mu\nu}D_{\mu\nu} + \beta^{\psi 4} \times K_{\mu\nu}^{\omega}D_{\mu\nu} \tag{3.5}$$

The super-scripting of coefficients $\alpha^{\psi 4}$ and $\beta^{\psi 4}$ was chosen to avoid confusion with the similarly named terms in Equation 3.2 while maintaining consistency with the variable names in the Psi4 codebase. The superscript in $K_{\mu\nu}^{\omega}$ is a label and not an exponential power. Notice that neither of these exchange matrices is termed as either the long-range

or short range exchange matrix. Most authors follow the naming convention of Savin in associating the error function with short range terms and the complementary error function with the long-range terms. In our working equation, Equation 3.5, the complementary error function is missing, so calling either term a "long-range" matrix would be inconsistent with the literature as would be the accompanying tacit suggestion that the HF exchange matrix is *the* long-range exchange term.

3.0.2 DF Approximation to DFT HF Exchange

To motivate the form of our working equation, recall that in density fitting, the electron density $\rho_{\mu\nu}$ in the ERI $(\mu\nu|\lambda\sigma)$ was approximated by sums of scaled orbitals χ^Q in an auxiliary basis set scaled by coefficients $d^Q_{\mu\nu}$ as below

$$(\mu\nu|\lambda\sigma) \approx d^Q_{\mu\nu}(Q|\lambda\sigma)$$

where the form of the coefficients $d^Q_{\mu\nu}$ was found to be $(\mu\nu|P)[J^{-1}]_{PQ}$[1],[2]. Then apply the same approximation to the electron density $\rho_{\mu\nu}$ in the integral $(\mu\nu|\omega|\lambda\sigma)$ to obtain:

$$(\mu\nu|\omega|\lambda\sigma) \approx (\mu\nu|P)[J^{-1}]_{PQ}(Q|\omega|\lambda\sigma)$$

where the operator ω in $(\mu\nu|\omega|\lambda\sigma)$ can be understood as the error function or the complimentary error function with extinction coefficient ω. This allows us to deduce our working equation for the Psi4 $K^{\omega}_{\mu\nu}$ as:

$$K^{\omega\psi_4}_{\mu\nu} = C_{\lambda a}(\mu\lambda|P)[J^{-1}]_{PQ}(Q|\omega|\nu\sigma)C_{\sigma a} \tag{3.6}$$

This term is calculated in Psi4 according to Algorithm 5. Once it is calculated, it is used to calculate each orbital's contribution to the exchange energy, and the inclusion of this term by Psi4 is also demonstrated in Algorithm 5.

Algorithm 5 Contractions to form DF ${}^{\omega}K_{\mu\nu}$

1: Initialize $(\mu\nu|\omega|P)$ ▷ $O(N_{bf}^2 N_{aux})$`mints` calculations
2: Initialize $(\mu\nu|P)$ ▷ $O(N_{bf}^2 N_{aux})$`mints` calculations
3: Contract $W_{\mu\nu}^{Q} = (\mu\nu|P)[J^{-1}]_{PQ}$ ▷ $O(N_{bf}^2 N_{aux}^2)$ `GEMM` ops
4: **for** SCF Iterations **do**
5: Contract $X_{\mu a}^{Q} = W_{\mu\lambda}^{Q} C_{\lambda a}$ ▷ $O(N_{occ} N_{bf}^2 N_{aux})$ `GEMM` ops
6: Contract ${}^{\omega}X_{\nu a}^{Q} = (\nu\sigma|\omega|P)C_{\sigma a}$ ▷ $O(N_{occ} N_{bf}^2 N_{aux})$ `GEMM` ops
7: Contract ${}^{\omega}K_{\mu\nu} = X_{\mu a}^{Q}\ {}^{\omega}X_{\nu a}^{Q}$ ▷ $O(N_{occ} N_{bf}^2 N_{aux})$ `GEMM` ops
8: $E_x \mathrel{+}= \beta^{\psi_4} \times {}^{\omega}K_{\mu\nu} \cdot D_{\mu\nu}$

3.1 Range Separated DFT with `DFHelper`

For calculating HF exchange matrices, a naïve implementation of the density fitting implementation will require at least $O(N_{bf}^2 * N_{aux})$ space for intermediates $(Q|\mu\nu)$ used only in the transformation $B_{\mu\nu}^{Q} = [J^{-\frac{1}{2}}]_{QP}(P|\mu\nu)$ during the initialization of the DFHF method. Memory requirements for such intermediate terms can be mitigated in the two following ways. We can follow an in-place transformation by transforming $(P|\mu\nu) \rightarrow B_{\mu\nu}^{Q}$ in pieces into a buffer then copying back into the memory in which $(P|\mu\nu)$ was initialized. Alternatively, we can have an initialization buffer for $(P|\mu\nu)$ and transform directly into the final address of $B_{\mu\nu}^{Q}$. The former is problematic because the copy step is a bottle-neck in parallel environments where there are more processing cores than memory channels. A large contribution of `DFHelper` was to implement the latter transformation workflow, and it is discussed at length in Scheiber's thesis. The purpose of this section is to discuss the implementation of Algorithm 6 in `DFHelper` .

Another contribution to `DFHelper` in Scheiber's thesis is the use of Schwarz Screening to the treatment of integrals. Schwarz Screening can lead to significant cost savings in QC calcuations both in terms of cost and time. It is then desirable to us to be able to extend the cost savings gained for normal HF calculations to range-separated DFT calculations. Discussing the validity of such an extension would consist of recognizing that the sparsity pattern offered by Schwarz screening on regular three-index ERI's might work for range-separated DFT as well. To do so, first recognize that the range-separated ERI $(P|\omega|\mu\nu)$

Algorithm 6 Conventional DF Range Separated DFT Coulomb and Exchange Term Workflow

1: Populate $(\mu\nu|P)$ ▷ $O(N_{bf}^2 \times N_{aux})$ `mints` calculations
2: Populate $(\mu\nu|\omega|P)$ ▷ $O(N_{bf}^2 \times N_{aux})$ `mints` calculations
3: Contract $B_{\mu\nu}^Q = (\mu\nu|P)[J^{-\frac{1}{2}}]_{PQ}$ ▷ $O(N_{bf}^2 N_{aux}^2)$ `GEMM` ops
4: Contract $W_{\mu\nu}^Q = (\mu\nu|P)[J^{-1}]_{PQ}$ ▷ $O(N_{bf}^2 N_{aux}^2)$ `GEMM` ops
5: **for** i=1 to `SCF MaxIters` **do**
6: Get New Coefficients $C_{\lambda a}$ and Densities $D_{\lambda\sigma}$
7: Calculate $J_{\mu\nu} = B_{\mu\nu}^Q B_{\lambda\sigma}^Q D_{\lambda\sigma}$ ▷ Two $O(N_{bf}^2 N_{aux})$ `GEMV` ops
8: Calculate $K_{\mu\nu} = C_{\lambda a} B_{\mu\lambda}^Q B_{\nu\sigma}^Q C_{\sigma a}$ ▷ Two $O(N_{occ} N_{bf}^2 N_{aux})$ `GEMM` ops
9: Calculate ${}^{\omega}K_{\mu\nu} = C_{\lambda a} W_{\mu\nu}^Q (Q|\omega|\nu\sigma) C_{\sigma a}$ ▷ Three $O(N_{occ} N_{bf}^2 N_{aux})$ `GEMM` ops
10: Diagonalize Fock Matrix and Calculate Energies

is an inner product between the function pair $\mu\nu$ and the pair composed of the auxiliary function P and the ω dampening function as the normal ERI $(P|\mu\nu)$ is an inner product between the pair $\mu\nu$ and the auxiliary function P. Then, recall that Schwarz screening consists of ignoring all ERI's that are less than some tolerance scaled by the maximum integral over all elements in some basis. The form of the ω dampening function implies that the maximum ERI $(P|\mu\nu)$ is an upper bound for the maximum ERI $(P|\omega|\mu\nu)$. This discussion does not necessarily necessitate that the sparsity patterns would be exactly the same for the two kinds of integrals, but they do offer some explanation as to why our initial tests did not show a great difference in energies between range separated calculations using the sparsity pattern and not using the sparsity pattern.

The reasoning for opening a discussion of sparsity patterns is that some HF kernels in Psi4 outside of `DFHelper` perform Schwarz Screening by copying integrals $B_{\mu\nu}^Q$ into a sparse buffer $B_{\mu\nu^\mu}^Q$. That this amount of copying takes place in every HF iteration causes a similar bottleneck to the one in in the metric contraction discussed above. The treatment of sparsity from Schwarz Screening in `DFHelper` does not necessitate such a transform as the ERI values $B_{\mu\nu}^Q$ are stored as "sparse" which is to say that neglected terms are not calculated or kept in memory. To have the proper indices for dense linear algebra kernels, the SCF coefficients $C_{\lambda a}$ are pruned as $C_{\lambda a} \rightarrow C_{\lambda^\mu a}$. This transform scales as $O(N_{occ} \times N_{bf}^2)$ for an entire SCF iteration in time and number of memory movements, but because the

spatial requirements are only $O(N_{occ} \times N_{bf})$, the serial bottleknecks that might be similar to the one above are substantially mitigated. Savings from using `DFHelper` are discussed in Sheiber's thesis as well, but the storage and transformation patterns in `DFHelper` were not available for range-separated DFT calculations.

Extending the technology in `DFHelper` to treat range separated DFT exchange consisted of expanding the initialization code and the Fock matrix building code to accommodate all the terms in Equation 3.6 following the steps in Algorithm 6. The initialization code in `DFHelper` was expanded to to include construction of the tensor $(Q|\omega|\mu\nu)$ and the metric contraction $(\mu\nu|P)[J^{-1}]_{PQ}$ along the ERI sparsity patterns in `DFHelper` . Code was added to the Fock Matrix construction functions in `DFHelper` to perform the transformation of the the terms $W^Q_{\mu\nu} = (\mu\nu|P)[J^{-1}]_{PQ}$ and $(\mu\nu|\omega|Q)$ from Equation 3.6 against the SCF orbital coefficients then contracting the transformed orbitals against each other to form $K^{\omega}_{\mu\nu}$. These transformations and initializations are shown in Algorithm 5 and were discussed in an earlier section. The sparsity machinery used in preforming these transformations as well as the memory layout used were discussed in Schieber's thesis. Inclusion of all these transformations leads to a `DFHelper` object that performs all the steps included in Algorithm 6.

3.1.1 Results

The major improvements seen from the change to handling exchange and Coulomb matrices in Hartree-Fock calculations in `DFHelper` are seen and magnified in our first round of additions to that library. Timings are presented in Figure 3.1.

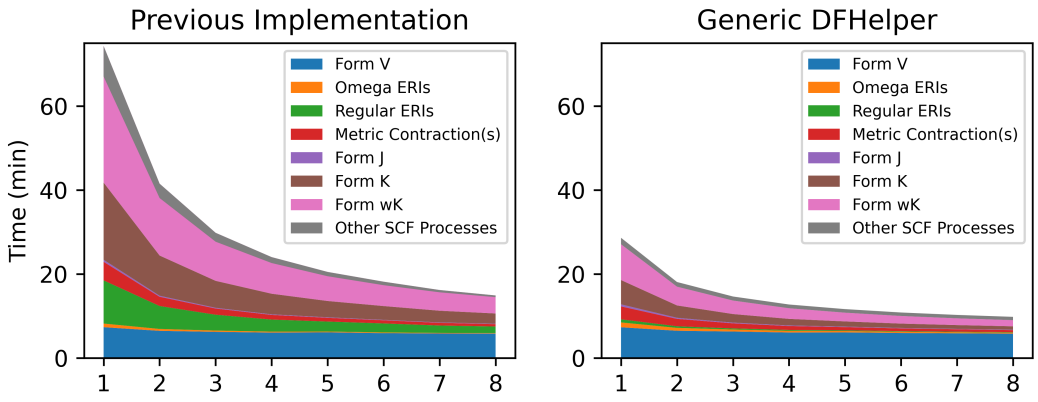

Figure 3.1: Timings of DFT Calculations of a Drug Fragment with 1206 Orbital Basis Functions on a Intel(R) Core(TM) i7-9800X CPU @ 3.80GHz over Four Memory Channels.

The `Form V` timer in Figure 3.1 represents DFT grid evaluation and the subsequent calculation of physical terms dependent on that grid. It becomes a more significant portion of the total DFT process as more cores are added, however, the work taken to evaluate it is $O(N_{bf}^3)$. The difference between the times for constructing Fock Matrix-like components in the previous implementation in Psi4 and the implementation in `DFHelper` is magnified relative to the analogous difference in plain HF calculations by the fact that there are more exchange terms to evaluate in this problem. That is to say that the reasons we see the improvement going from the previous implementation to a generic `DFHelper` implementation have already adequately been explained by Scheiber in his thesis and briefly discussed in our introcudtion to this section.

3.2 Combined Computation of Exchange Terms

There are cases where it is necessary to have separately termed DFT and HF exchange matrices for a DFT application. For the other cases, where only the DFT energy is needed, we can consider the formulation in Equation 3.5:

$$E_x = \alpha^{\psi 4} \times K_{\mu\nu} D_{\mu\nu} + \beta^{\psi 4} \times K^{\omega}_{\mu\nu} D_{\mu\nu}$$

and apply the distributive property of vector dot products to arrive at the formulation:

$$E_x = (\alpha^{\psi 4} \times K + \beta^{\psi 4} \times K^{\omega})_{\mu\nu} D_{\mu\nu} \tag{3.7}$$

where we have moved the subscripts outside the parentheses to emphasize that there is one matrix on the inside. The difference between evaluating the right hand sides in either equation only affects a computational prefactor. However, both matrices $K_{\mu\nu}$ and ${}^{\omega}K_{\mu\nu}$ are formed from ERI tensors of the same size and shape using the same density matrix. We can expand out their formulations to deduce a working equation for a more parsimonious approach to evaluating terms in the problem. Starting from Equation 3.5:

$$\begin{aligned}
E_x =& \alpha^{\psi 4} \times K_{\mu\nu} D_{\mu\nu} + \beta^{\psi 4} \times {}^{\omega}K_{\mu\nu} D_{\mu\nu} \\
=& \alpha^{\psi 4} C_{\lambda a} (\mu\lambda|P)(P|Q)^{-1}(Q|\nu\sigma) C_{\sigma a} D_{\mu\nu} + \beta^{\psi 4} C_{\lambda a} (\mu\lambda|\omega|P)(P|Q)^{-1}(Q|\nu\sigma) C_{\sigma a} D_{\mu\nu} \\
=& C_{\lambda a} (\alpha^{\psi 4} (\mu\lambda|P) + \beta^{\psi 4} (\mu\lambda|\omega|P))(P|Q)^{-1}(Q|\nu\sigma) C_{\sigma a} D_{\mu\nu}
\end{aligned}$$

we arrive at a formulation for the DFT E_x term that only involves returning one matrix from the Fock-type exchange kernel. Although for even two cores, the HF exchange formation is not the largest or even second-largest component of the total DFT SCF calculation time, it does require $O(N_{occ} N_{bf} N_{aux})$ effort, that is $O(n^4)$ where n is take simply as the number of electrons in a molecule, per iteration, so it can be expected to be a significant reduction in computational effort. As far as formulating this change goes, simply change line 2 in Algorithm 6 to include both $(\mu\nu|P)$ and $(\mu\nu|\omega|P)$ then eliminate line 8 in the same algorithm and included the combined ERI tensor in line 9. Such leaves us with Algorithm 7

Algorithm 7 Combined DF Range Separated DFT Coulomb and Exchange Term Workflow

1: Populate $(\mu\nu|P)$ ▷ $O(N_{bf}^2 \times N_{aux})$ `mints` calculations
2: Populate $\alpha^{\psi_4}(\mu\nu|P) + \beta^{\psi_4}(\mu\nu|\omega|P)$ ▷ $O(N_{bf}^2 \times N_{aux})$ `mints` calculations
3: Contract $B_{\mu\nu}^Q = (\mu\nu|P)[J^{-\frac{1}{2}}]_{PQ}$ ▷ $O(N_{bf}^2 N_{aux}^2)$ `GEMM` ops
4: Contract $W_{\mu\nu}^Q = (\mu\nu|P)[J^{-1}]_{PQ}$ ▷ $O(N_{bf}^2 N_{aux}^2)$ `GEMM` ops
5: **for** i=1 to `SCF MaxIters` **do**
6: Get New Coefficients $C_{\lambda a}$ and Densities $D_{\lambda\sigma}$
7: Calculate $J_{\mu\nu} = B_{\mu\nu}^Q B_{\lambda\sigma}^Q D_{\lambda\sigma}$ ▷ Two $O(N_{bf}^2 N_{aux})$ `GEMV` ops
8: Form ${}^{\omega}K_{\mu\nu} = C_{\lambda a} W_{\mu\nu}^Q (\alpha^{\psi 4}(Q|\nu\sigma) + \beta^{\psi_4}(Q|\omega|\nu\sigma)) C_{\sigma a}$ ▷ Three $O(N_{occ} N_{bf}^2 N_{aux})$ `GEMM` ops
9: Diagonalize Fock Matrix and Calculate Energies

3.2.1 Results and Comparison to Previous Implementation

As discussed above, we would expect the improvement over the two-exchange `DFHelper` formulation to be modest relative to the improvement relative to the previous implementation in Psi4. Indeed, as we can see in Figure 3.2, by the time the calculation is run on eight cores, the majority of the time in the calculation is spent outside the Fock-type kernels even before the addition of this change. The merits of using eight processing cores on this calculation even come into question because of the drop in overall efficiency due to the lack of parallelism in the other kernels involved in the DFT calculation.

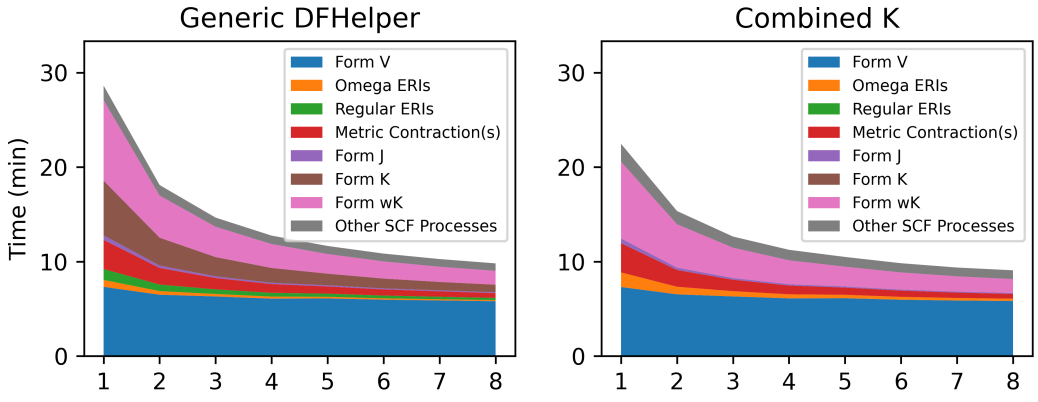

Figure 3.2: Timings of DFT Calculations of a Drug Fragment with 1206 Orbital Basis Functions on a Intel(R) Core(TM) i7-9800X CPU @ 3.80GHz over Four Memory Channels.

As discussed in Chapter 2, it is sometimes appropriate to use a different algorithm or to allocate resources differently depending on the setting with relevant conditions including the hardware and the reason that the calculation is being run. We remember that in that case, when it was necessary to run non-dampened DFT or a HF calculation on a single molecule on a workstation with a large number of cores, the `DirectDFJK` code was more appropriate, but when a large number of calculations were run on shared resources, it was more appropriate to use a pre-existing disk-based algorithm. In the current case, our algorithm performs better in general, however, the consideration of diminishing returns for increased use of shared resources does change the way in which a decision to allocate more resources to individual calculations is made. Figure 3.3 shows some timings for the previous implementation to help with these considerations.

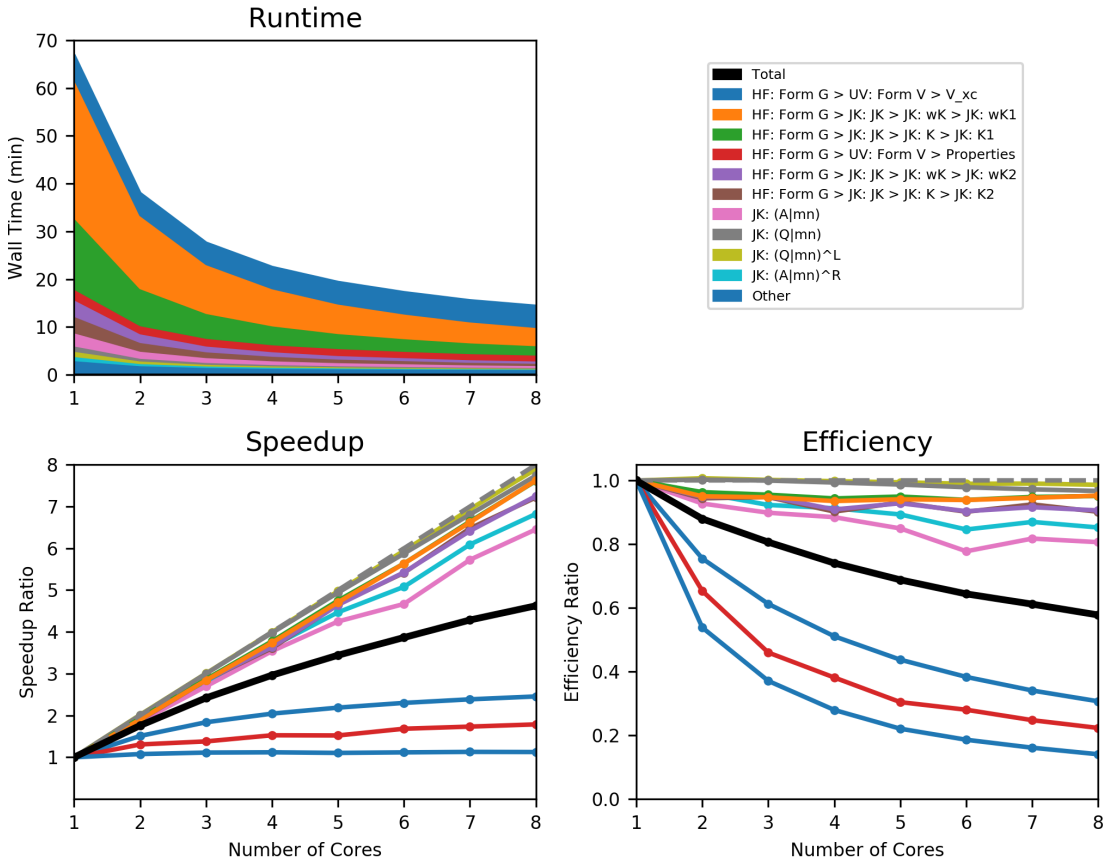

Figure 3.3: Timings of DFT, Parallel Efficiencies, and Speed-ups for Calculations Using The Previous Implementation of a Drug Fragment with 1206 Orbital Basis Functions on a Intel(R) Core(TM) i7-9800X CPU @ 3.80GHz over Four Memory Channels.

In the previous figure, we can see that because of the great expense of formation of the Fock-matrix component-like matrices which largely consist of GEMM operations, the DFT calculations have a parallel efficiency at around 80%. For these calculations, tasking more cores to a single-point energy is a minor consideration because of the efficiency of the parallelism being used in that calculation. However, consulting Figure 3.4, we can see that because the calculation of the DFT energy is now dominated by the grid formation and properties (times labeled V_xc and Properties in Figure 3.4) which are largely unparallelized.

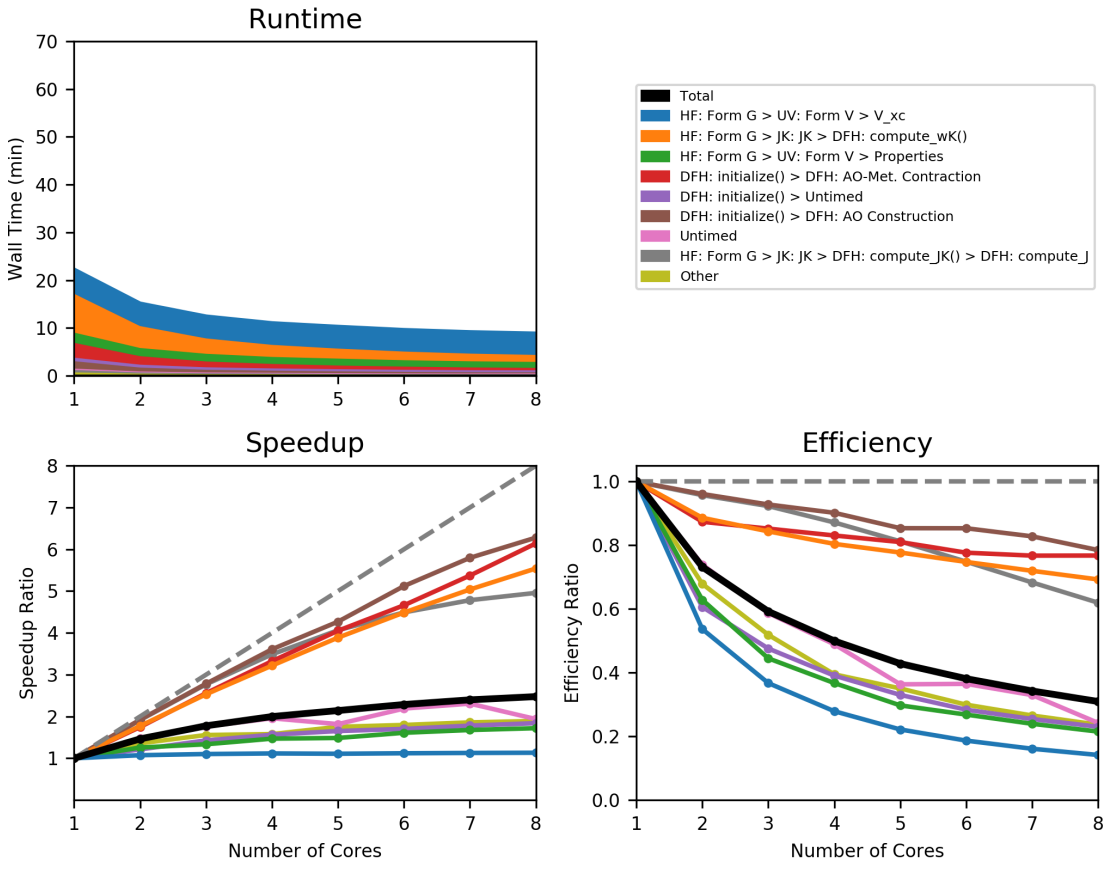

Figure 3.4: Timings of DFT, Parallel Efficiencies, and Speed-ups for Calculations Using the Combined Fomrulation of a Drug Fragment with 1206 Orbital Basis Functions on a Intel(R) Core(TM) i7-9800X CPU @ 3.80GHz over Four Memory Channels for comparison with Figure 3.3.

To discuss allocation of resources directly, four cores are needed for a full two-times speed-up with the combined formulation, and the more parallel-wise efficient algorithm is more costly. Because many of the applications of DFT rely on running a large number of calculations on numerous molecules, the most important consideration for performance becomes efficient use of core-hours because the results of a single-point energy only matter as much as all the other necessary energies. In this setting, with a limited number of computational resources, it becomes crucial to fit as many single point calculations onto a single node as possible to approach pleasant parallelism bearing in mind that to run a calculation out of core will require us to use the previous implementation in Figure 3.1.

3.3 Resolution of the identity for Coulomb Matrix Memory Cost Reduction

3.3.1 Motivation and Formulation

At this point, our `ωCombine` algorithm calculates two matrices:

$$J_{\mu\nu} = (\mu\nu|P)[J^{-\frac{1}{2}}]_{PQ}[J^{-\frac{1}{2}}]_{QR}(R|\lambda\sigma)D_{\lambda\sigma}$$
$$K^{\omega}_{\mu\nu} + K_{\mu\nu} = C_{\lambda a}((\mu\lambda|P)[J^{-1}]_{PQ})((Q|\nu\sigma) + (Q|\omega|\nu\sigma))C_{\sigma a}.$$

This formulation achieves great savings on the iteration time for Kohn-Sham (KS) DFT calculations. We, however, are holding three ERI tensors in order to compute these terms:

$$(\mu\nu|P)[J^{-\frac{1}{2}}]_{QP}$$
$$(\mu\nu|P)[J^{-1}]_{QP}$$
$$(\mu\nu|P) + (\mu\nu|\omega|Q)$$

This formulation requires two $O(N_{bf}^2 N_{aux})$ ERI tensor populations for the $(\mu\nu|P)$ term and the $(\mu\nu|P) + (\mu\nu|\omega|P)$ term besides two separate $O(N_{bf}^2 N_{aux}^2)$ metric contractions: $(\mu\nu|P)[J^{-\frac{1}{2}}]_{QP}$ and $(\mu\nu|P)[J^{-1}]_{QP}$ each of which can require more computation than an

entire HF iteration. Beyond computational costs, each of the intermediates we've discussed requires $O(N_{bf}^2 N_{aux})$ storage space which can push a calculation into using a disk-based algorithm and into facing all the accompanying serial bottlenecks. Beyond needing to acces disk tensors, these bottlenecks require a Psi4 user to use the previous implementation referenced in Figure 3.1 and Figure 3.3, more than doubling the resources consumed by each DFT calculation. A reformulation of these matrices that would require less memory would greatly reduce the cost of calculations that would otherwise require disk use and would also allow more calculations to fit in core on a given node as recommended in section 3.2. If we change our formulation of the Coulomb matrix $J_{\mu\nu}$ using a resolution of the identity as below:

$$\begin{aligned} J_{\mu\nu} &= (\mu\nu|P)[J^{-\frac{1}{2}}]_{PQ}[J^{-\frac{1}{2}}]_{QR}(R|\lambda\sigma)D_{\lambda\sigma} \\ &= (\mu\nu|P)[J^{-1}]_{PQ}(Q|\lambda\sigma)D_{\lambda\sigma} \\ &= (\mu\nu|P)[J^{-1}]_{PR}[J]_{RS}[J^{-1}]_{SQ}(Q|\lambda\sigma)D_{\lambda\sigma} \end{aligned} \tag{3.8}$$

We obtain a formulation of the Coulomb and exchange matrices that only involves the ERI tensors:

$$(\mu\nu|P)[J^{-1}]_{QP}$$

$$(\mu\nu|P) + (\mu\nu|\omega|Q)$$

Which means that we can avoid the metric contraction $(\mu\nu|P)[J^{-\frac{1}{2}}]_{QP}$ and that our Coulomb and exchange matrix determinations require two thirds the amount of space of Algorithm 7.

These calculations are presented in Algorithm 8

Algorithm 8 Combined DF Range Separated DFT Coulomb and Exchange Term Workflow

Populate $(\mu\nu|P)$ ▷ $O(N_{bf}^2 \times N_{aux})$ `mints` calculations
Populate $\alpha^{\psi_4}(\mu\nu|P) + \beta^{\psi_4}(\mu\nu|\omega|P)$ ▷ $O(N_{bf}^2 \times N_{aux})$ `mints` calculations
Contract $W_{\mu\nu}^{Q} = (\mu\nu|P)[J^{-1}]_{PQ}$ ▷ $O(N_{bf}^2 N_{aux}^2)$ `GEMM` ops
for i=1 to `SCF MaxIters` **do**
    Get New Coefficients $C_{\lambda a}$ and Densities $D_{\lambda\sigma}$
    Calculate $J_{\mu\nu} = W_{\mu\nu}^{Q}(Q|P)W_{\lambda\sigma}^{P}D_{\lambda\sigma}$
        ▷ Two $O(N_{bf}^2 N_{aux})$ `GEMV` ops & One $O(N_{aux}^2)$ LU Evaluation
    Form ${}^{\omega}K_{\mu\nu} = C_{\lambda a}W_{\mu\nu}^{Q}(\alpha^{\psi 4}(Q|\nu\sigma) + \beta^{\psi_4}(Q|\omega|\nu\sigma))C_{\sigma a}$
        ▷ Three $O(N_{occ}N_{bf}^2 N_{aux})$ `GEMM` ops
    Diagonalize Fock Matrix and Calculate Energies

3.3.2 Results

There are two benefits of this reformulation to consider: elimination of a metric contraction and reduced memory cost. Eliminating one of the metric contractions (that is, halving the red section in Figure 3.5) offers a slight improvement in performance that becomes more modest as more cores are assigned to a calculation (Figure 3.5.)

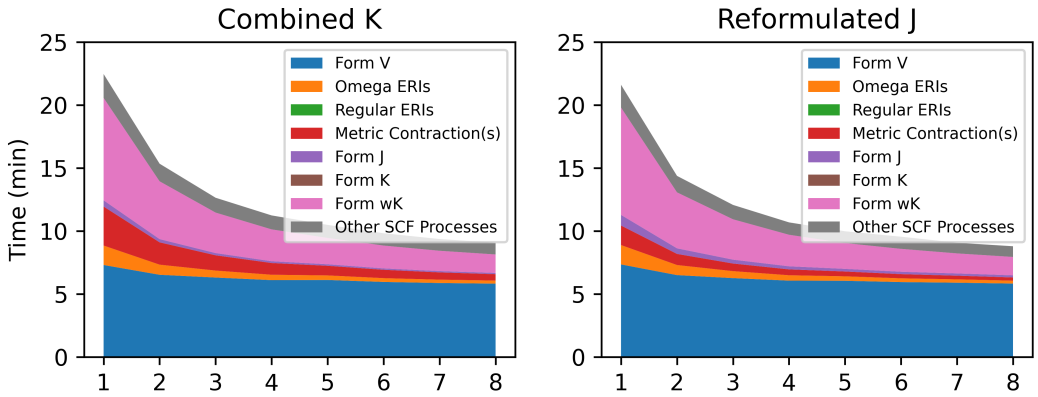

Figure 3.5: Timings of DFT Calculations of a Drug Fragment with 1206 Orbital Basis Functions on a Intel(R) Core(TM) i7-9800X CPU @ 3.80GHz over Four Memory Channels with the formulation from Equation 3.7 (left) and Equation 3.8 (right)

Table 3.1: Changes in metric contraction times going from one to eight cores in each of the two formulations for the Coulomb matrix. Memory costs for the same calculation in `DFHelper` with a Schwarz cutoff of 1.0×10^{-12}.

Calculation	3 Tensors 1 Core	3 Tensors 8 Cores	2 Tensors 1 Core	2 Tensors 8 Cores
Calculation Wall Time	1347.93s	544.39s	1296.75s	527.50s
Metric Contraction Time	185.670s	30.227s	92.861s	15.076s
`DFHelper` Memory Cost	50.278 GiB	50.354 GiB	33.643 GiB	33.719 GiB

Beside including the modest improvement in calculation times from the elimination of a Coulomb Metric contraction, Table 3.1 shows the two thirds memory cost reduction suggested earlier.

Because increasing the performance of the code and reformulating the Fock Matrix Components can only reduce the calculation time of procedures that comprise less than half the calculation time when range-seperate hybrid DFT calculations are performed on more than one core, a change such as this represents a significant cost reduction for batches of calculations that otherwise would not have been able to be run on a single node. To convey the effect that reducing the memory cost in a large dataset can have, some costs of `DFHelper` DFT calculations in some organic molecules are plotted in Figure 3.6.

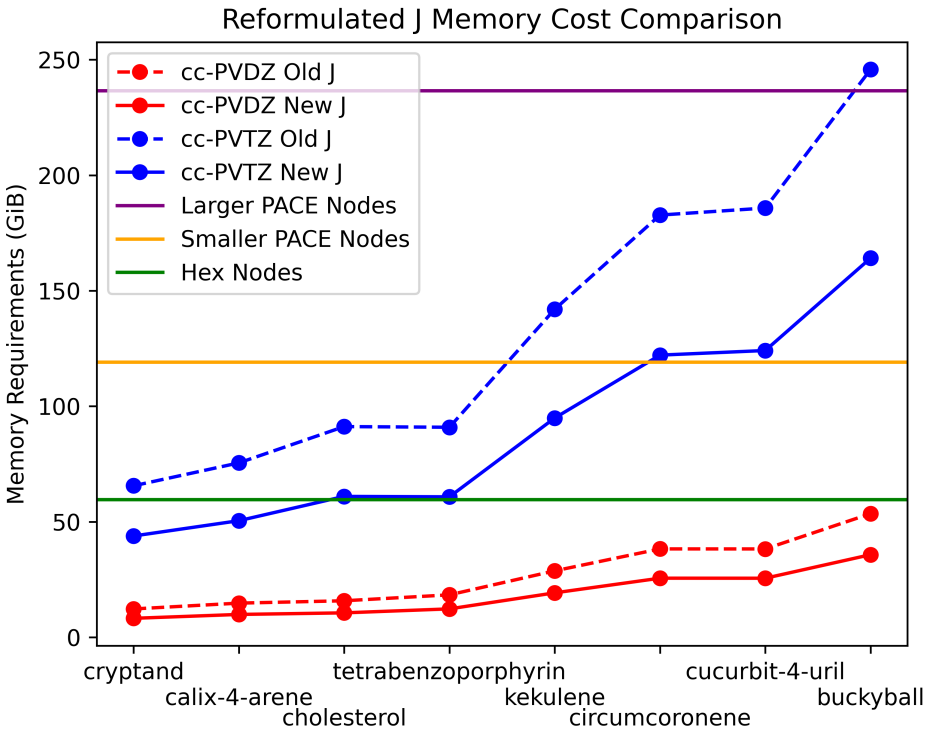

Figure 3.6: Graph showing the memory costs of various biomolecules in `DFHelper` using different basis sets and different formulations of the Coulomb and Exchange Matrices. The molecules are arranged from smallest (left) to largest (right) in terms of memory cost. A Schwarz cutoff of 1.0×10^{-12}.

The Schwarz cutoff is mentioned in Table 3.1 and Figure 3.6 because the ERI tensors used in these calculations are stored sparse, so the memory costs of these calculations are dependent on the amount of screening involved.

CHAPTER 4
MULTIBASIS FOCK MATRIX COMPONENT-LIKE OBJECTS FOR SAPT-F12 TERMS

4.1 Exposition of Terms and Choice of Working Equations

The HF energy fails to include electron correlation. That is to say that the Roothan Equation:

$$F(C)C = EC \tag{4.1}$$

where F is the Fock Matrix, C is the orbital coefficient matrix for the basis in which the molecule is described, and E is the diagonal matrix of eigenvalues of the Fock matrix, is based on the mean field approximation which treats the interactions between electrons as being between each electron and a mean field being "shown" by the others. This approximation then fails to account for the effect that the specific location of a given electron has on the motion of another electron. The difference between the HF energy and the "true" energy of a molecule is the correlation energy. It can be calculated using many methods including DFT, coupled cluster theory, the configuration interaction, and perturbation theory. All the wavefunction methods among these theories (that is, all the listed theories that are not DFT) deal with multiple Slater determinants[5]. One difficulty with these wavefunction-based electron correlation methods is that they are slowly convergent with respect to the basis set employed[18],[19]. An approach to accelerate convergence with respect to basis set is to use 'explicitly correlated' methods, which introduce additional terms into the wavefunction that depend linearly on the inter-electronic distance r_{12}, or functions of this distance[20]. The latter are generically called F12 methods, and they have been reviewed extensively[21]. The Patkowski group at Auburn University, notably including Dr. Kodrycka, was working on F12 formulations of Symmetry Adapted Perturbation Theory

Table 4.1: Types of bases used in SAPT-F12 spanned by various tabulated indexing variables.

Atomic Orbitals	Atomic			Auxiliary	
Indices	$\mu, \nu, \ldots$			$\mu', \nu', \ldots$	
Molecular Orbitals	Occupied	Virtual	Any Molecular	CABS	RI Basis
Indices	$i, j, \ldots$	$a, b, \ldots$	$r, s, \ldots$	$x, y, \ldots$	$\alpha, \beta, \ldots$

(SAPT), which computes the interaction energy between two molecules.[22] Their group was interested in improving the efficiency of their code and requested the generalization of the Fock matix builds to accomodate the particular Fock-like matrices that enter into F12 theories. This Chapter discusses my efforts to develop such code.

To list some of the terms involved with SAPT-F12 theory, we have borrowed notation from Werner *et al.*[23] which is stated in Table 4.1. This theory seeks to treat an interaction space that includes some explicitly correlated terms[21]:

$$|\Phi_{ij}^{kl}\rangle = |\Phi_{ij}^{\alpha\beta}\rangle \mathcal{F}_{\alpha\beta}^{kl} \tag{4.2}$$

where the terms: $\mathcal{F}_{\alpha\beta}^{kl} = \langle kl|\hat{F}_{12}\hat{Q}_{12}|\alpha\beta\rangle$ are coefficients that help define the magnitude of the contributions of the terms $|\Phi_{ij}^{\alpha\beta}\rangle$ to some electron configuration. There are many choices of correlation factors $\hat{F}_{12}$ which are the domain of other work. The operator $\hat{Q}_{12}$ is needed to enforce strong orthogonality of the explicitly correlated functions in Equation 4.2. Obtaining the functions in this space entails calculating the energy from the Hylleraas functional[24]:

$$E_2 = \langle \Psi^{(1)}|\hat{H}^{(0)} - E^{(0)}|\Psi^{(1)}\rangle + 2\langle \Psi^{(1)}|\hat{H}|\Psi^{(0)}\rangle \tag{4.3}$$

This term requires computing the intermediates:

$$A_{kl,mn} = U_{\bar{kl},mn}^{F} + F_{\bar{kl},mn}^{2} + F_{k\bar{l},mn}^{2} - \tilde{Y}_{k\bar{l},mn} - \bar{F}_{\alpha\beta}^{kl} P_{\alpha\beta} F_{\alpha\beta}^{mn} \tag{4.4}$$

in which the terms $F^{kl}_{\bar{x}i}$, $F^{kl}_{x\bar{i}}$ (enumerated below) contain terms $f_{\phi\gamma}$ and $k_{\phi\gamma}$ which are relevant to this thesis. The terms U, $\tilde{Y}$, and P are not enumerated because their calculation is not discussed in this thesis.

$$F^{kl}_{\bar{x}i} = (f_{xr} + k_{xr})F^{kl}_{ri} + (f_{xy} + f_{xy})F^{kl}_{yi} \quad \& \quad F^{kl}_{x\bar{i}} = F^{kl}_{xr}(f_{ri} + k_{ri}) + F^{kl}_{xy}(f_{yi} + k_{yi})$$

as well as the other intermediate:

$$C^{kl}_{ab} = f_{ax}F^{kl}_{xb} + F^{kl}_{ax}f_{xb} \tag{4.5}$$

Within these terms, we are interested in calculating components of the terms $f_{\phi\rho}$ (where the indexing variables ϕ and ρ are chosen because they are generic to the terms in Table 4.1.) Each of these matrices $f_{\phi\rho}$ is a Fock-type matrix which has its elements calculated from an operator in much the same way as the Fock matrix. This operator is composed of terms corresponding to the kinetic energy, nucleus-electron electrostatics, electron-electron coulomb interactions, and electronic exchange as: $\hat{f} = \hat{t} + \hat{v} + 2\hat{j} + \hat{k}$ where the terms much like those in Equation 1.4 and Equation 1.5. The only difference between the matrix elements of these two operators and those of the conventional Coulomb and exchange operators in Equation 1.4 and Equation 1.5 is that the indices correspond to elements of bases other than the atomic orbitals that describe a molecule in HF theory.

In our case, we will be optimizing the calculation of DF approximations to Coulomb and Exchange-like terms $J_{\phi\rho}$ and $K_{\phi\rho}$ where ϕ and γ index over either the AO or a union basis used for calculations involving the Complementary Auxiliary Basis Set (CABS)[25] in Psi4. The union basis contains the AO basis as a subset as well as another auxiliary basis. We have cited libraries in Psi4 that offer optimized construction of Coulomb and exchange matrices, but these libraries encode two symmetries in their formulations of $J_{\mu\nu}$ and $K_{\mu\nu}$ that are broken in our formulations of the terms $J_{\phi\rho}$ and $K_{\phi\rho}$. The first of these is that the orbital densities $D_{\mu\nu}$ used to calculate the Fock matrix components are in the same basis as

the Fock components themselves. In our case, the density matrix will be in the AO basis, and the Fock components will be: AO by AO (the conventional case), AO by union, and union by union. The other symmetry is that the conventional Fock components are square. These conditions require us to re-encode some of the modules used in the formulations of these matrices and to reformulate these matrices from the choices made in Equation 1.11 and in Equation 1.12.

4.2 Reformulation

In this section, we will discuss the formulations of the new matrices and the changes made to the code in order to calculate them. Different from the HF case where there are two matrices to consider, we have six matrices to consider: J_{xy}, K_{xy}, $J_{x\mu}$, $K_{x\mu}$, $J_{\mu\nu}$, and $K_{\mu\nu}$. We are only interested in forming each of these matrices using the DF approximation. A naïve extension of the conventional working equations Equation 1.11 and Equation 1.12, adapting our usual notation for the terms appropriate to this theory, will give us the following six formulations with a change in notation to that in Table 4.1:

$$\begin{aligned} J_{xy} &= B^{\mu'}_{xy} B^{\mu'}_{\lambda\sigma} D_{\lambda\sigma} \\ K_{xy} &= C_{\lambda i} B^{\mu'}_{x\lambda} B^{\mu'}_{y\sigma} C_{\sigma i} \\ J_{x\mu} &= B^{\mu'}_{i\mu} B^{\mu'}_{\lambda\sigma} D_{\lambda\sigma} \\ K_{x\mu} &= C_{\lambda i} B^{\mu'}_{x\lambda} B^{\mu'}_{\mu\sigma} C_{\sigma i} \\ J_{\mu\nu} &= B^{\mu'}_{\mu\nu} B^{\mu'}_{\lambda\sigma} D_{\lambda\sigma} \\ K_{\mu\nu} &= C_{\lambda i} B^{\mu'}_{\mu\lambda} B^{\mu'}_{\nu\sigma} C_{\sigma i} \end{aligned} \tag{4.6}$$

where the ERI tensors with the same data have been given the same color, and terms $B^{\mu'}_{\phi\gamma}$ are analogous to terms in Equation 1.11 and Equation 1.12 and are not meant to be analogous to other terms from the Perturbation theory literature. Populating them all separately will require added work if done naïvely.

Algorithm 9 Naive Construction of Two Basis Fock Components

1: Initialize $(xy\|\mu')$	$\triangleright\, O(N_{aux}N_{CABS}^2)$ `mints` evaluations
2: Initialize $(x\mu\|\mu')$	$\triangleright\, O(N_{bf}N_{aux}N_{CABS})$ `mints` evaluations
3: Initialize $(\mu\nu\|\mu')$	$\triangleright\, O(N_{bf}^2N_{aux})$ `mints` evaluations
4: contract $B_{xy}^{\mu'} = (xy\|\nu')[J^{-\frac{1}{2}}]_{\nu'\mu'}$	$\triangleright\, O(N_{aux}^2N_{CABS}^2)$
5: contract $B_{x\mu}^{\mu'} = (x\mu\|\nu')[J^{-\frac{1}{2}}]_{\nu'\mu'}$	$\triangleright\, O(N_{bf}N_{aux}^2N_{CABS})$
6: contract $B_{\mu\nu}^{\mu'} = (\mu\nu\|\nu')[J^{-\frac{1}{2}}]_{\nu'\mu'}$	$\triangleright\, O(N_{bf}^2N_{aux}^2)$
7: construct $J_{xy} = B_{xy}^{\mu'}B_{\mu\nu}^{\mu'}D_{\mu\nu}$	$\triangleright\, O(N_{aux}N_{CABS}^2 + N_{bf}^2N_{aux})$
8: construct $K_{xy} = C_{\mu i}B_{x\mu}^{\mu'}B_{y\nu}^{\mu'}C_{\nu i}$	$\triangleright\, O(N_{occ}N_{aux}N_{CABS}^2 + N_{occ}N_{bf}N_{aux}N_{CABS})$
9: construct $J_{x\mu} = B_{x\mu}^{\mu'}B_{\lambda\sigma}^{\mu'}D_{\lambda\sigma}$	$\triangleright\, O(N_{bf}N_{aux}N_{CABS} + N_{bf}^2N_{aux})$
10: construct $K_{x\mu} = C_{\nu i}B_{x\nu}^{\mu'}B_{\mu\lambda}^{\mu'}C_{\lambda i}$	$\triangleright\, O(N_{occ}N_{bf}^2N_{aux} + N_{occ}N_{bf}N_{aux}N_{CABS})$
11: construct $J_{\mu\nu} = B_{\mu\nu}^{\mu'}B_{\lambda\sigma}^{\mu'}D_{\mu\nu}$	$\triangleright\, O(N_{bf}^2N_{aux})$
12: construct $K_{\mu\nu} = C_{\lambda i}B_{\mu\lambda}^{\mu'}B_{\nu\sigma}^{\mu'}C_{\sigma i}$	$\triangleright\, O(N_{occ}N_{bf}^2N_{aux})$

One solution to that problem is to simply initialize a `JK` object using the CABS basis rather than the AO basis. These matrices can then be populated simply padding the matrices $C_{\lambda i}$ and $D_{\lambda\sigma}$ with zeros for all basis functions $\kappa \in$ union basis such that $\kappa \notin$ AO. This would solve the problem of multiple redundant data and would obviate the need to populate all these tensors. However, we shall see later that this formulation, an ingenious use of the code by Dr. Kodrycka, leads to such an amount of extra work that she suggested a reformulation to yield a significant improvement in speed.

4.2.1 Changes to the Naïve Workflow

Our goal in reformulating these Fock Matrix-like terms will be to avoid redundant and large calculations as well as to recycle data and intermediates when possible. To the extent that we can, we will reuse the machinery existing in `DFHelper` , but our changes are significant and pervasive enough to the `DFHelper` base class that a `C++ subclass` of the `DFHelper` class appeared advantageous for the implementation. We will discuss changes to the mathematical formulations of the individual terms, specific recycling of data, and the proper choices for Schwarz masks.

The primary trade off in forming the ERI tensors in such a way that `GEMM` operations

over orbital density matrices $D_{\mu\nu}$ and coefficient matrices $C_{\mu i}$ padded with $0's$ no longer take place is in memory versus time. There is no way to avoid forming separate $B^{\mu'}_{\mu\nu}$, $B^{\mu'}_{x\mu}$, and $B^{\mu'}_{xy}$ to avoid these transformations. It is generally true that because $N_{aux} > 2N_{bf}$ and $N_{union} = N_{bf} + N_{aux}$, the memory cost when adding the extra ERI arrays will be less than doubled, so the change is pursued.

Beyond filling the added storage needed for the new ERI tensors is the computational effort taken to fill them and the metric contractions taken to prepare them for use in the general workflow. Inspecting Algorithm 9, the terms associated with steps four, five, and six grow faster asymptotically that steps seven through 12, so these added metric contractions can easily cause the full workflow for the new formulation to take more time than that with the padded transformations in `DFHelper` . Note that that comes after remedying the problem of transforming over spurious zero matrix elements.

To mitigate these issues, avoid any recalculation of tensor elements. The first consideration to make is that the integrals $(\mu\nu|\mu')$ and $(x\mu|\mu')$ are all contained in $(xy|\mu')$, so the calculation of these tensors must be interleaved. This is to say that once an ERI term is calculated, it is saved in all the tensors where it is needed. In practice, only allocate memory for $(x\mu|\mu')$ and $(xy|\mu')$ because the calculation of $B^{\mu'}_{\mu\nu}$ is itself interleaved with the calculation of $B^{\mu'}_{x\mu}$. All these calculations can be reduced through use of Schwarz screening, but to avoid the access penalty incurred by checking multiple sparsity masks for each ERI computation, one mask is used for all three tensors $B^{\mu'}_{\phi\gamma}$. We chose the CABS mask because both the AO and auxiliary basis sets are subsets of the CABS basis.

Further considerations must be made to avoid calculating redundant terms between $B^{\mu'}_{x\mu}$ and $B^{\mu'}_{\mu\nu}$. `DFHelper` computes the metric contraction $B^{\mu'}_{\mu\nu} = (\mu\nu|\nu')[J^{-\frac{1}{2}}]_{\nu'\mu'}$ symmetrically which is to say that the element $B^{\mu'}_{\mu\nu}$ is calculated and then copied into $B^{\mu'}_{\nu\mu}$. Because $B^{\mu'}_{x\mu}$ does not share this symmetry for $x \in$ CABS, $x \notin$ AO basis, copying symmetric terms is not possible, and simply contracting $B^{\mu'}_{x\mu} = (x\mu|\nu')[J^{-\frac{1}{2}}]_{\nu'\mu'}$ then copying shared terms into $B^{\mu'}_{\mu\nu}$ will involve wasted work. To reduce the computational effort involved with all

these terms, it is crucial to compute for the index $\gamma \in$ CABS the full slice $B^{\mu'}_{\gamma\nu}$ when $\gamma \notin$ AO basis and only parts of the slice $B^{\mu'}_{\gamma\nu}$ such that $\nu \geq \tilde{\gamma}$ where $\tilde{\gamma}$ is the index for the function γ in the AO basis. We then copy these terms into the appropriate addresses in $B^{\mu'}_{x\mu}$ and $B^{\mu'}_{\mu\nu}$

The tensor $B^{\mu'}_{xy}$ (in red above) is the largest tensor our workflow demands, and it is also the only tensor that only gets used to calculate only one of our six terms. The other tensors get reused. Computing its metric contraction would be the longest component of our workflow in terms of time. Examining the conventional ERI Equation 1.14, we can see that it is not completely necessary to form contracted term $B^{\mu'}_{xy}$ if the uncontracted term $(\lambda\sigma|\nu')$ is available. Indeed, if we decompose the term $B^{\mu'}_{xy} = (xy|\mu')[J^{-\frac{1}{2}}]_{\mu'\nu'}$ then we can formulate the Coulomb term: $J_{xy} = (xy|\nu')[J^{-\frac{1}{2}}]_{\nu'\mu'}B^{\mu'}_{\lambda\sigma}D_{\lambda\sigma}$, allowing us to avoid the metric contraction: $B^{\mu'}_{xy} = (xy|\mu')[J^{-\frac{1}{2}}]_{\nu'\mu'}$ which is the most expensive term to calculate in this entire workflow without adding the tensor $(\mu\nu|\mu')$ to our memory costs.

These changes lead us to the workflow for calculating two-basis Fock-like components in Algorithm 10. Important to note is that this algorithm has no $O(N^2_{aux}N^2_{CABS})$ calculation steps, all $O(N^2_{bf}N_{aux})$ and $O(N_{bf}N_{aux}N_{CABS})$ `mints` evaluations have been included in the $O(N_{aux}N^2_{CABS})$ step, and the symmetry of the $O(N^2_{bf}N^2_{aux})$ contraction is still employed as that transformation is included in the $O(N_{bf}N^2_{aux}N_{CABS})$ contraction.

4.3 Results

Our results will concern the implementation of Algorithm 10 and its comparison to the exploitation of `DFHelper` that involves padding the density matrix and coefficient matrix with zeros. To assess the success the implementation of our algorithm, we will ensure that the algorithm achieves lower runtimes than the padded `DFHelper` code and that it parallelizes at least as well as that algorhtm.

We will first compare the wall times of the stages of Algorithm 10 to that of the padded SCF coefficient and density matrices in `DFHelper` on a benzene dimer, and the two nu-

Algorithm 10 Final Construction of Two Basis Fock Components

1:	Initialize $(xy\|\mu') \rightarrow (x\mu\|\mu') \rightarrow (\mu\nu\|\mu')$	$\triangleright O(N_{aux}N_{CABS}^2)$ `mints` evaluations
2:	Contract $B_{x\mu}^{\mu'} = (x\mu\|\nu')[J^{-\frac{1}{2}}]_{\nu'\mu'}$	$\triangleright O(N_{bf}N_{aux}^2)N_{CABS}$
3:	Copy $B_{x\mu}^{\mu'} \rightarrow B_{x\mu}^{\mu'} \rightarrow B_{\mu\nu}^{\mu'}$	$\triangleright O(N_{bf}N_{aux}N_{CABS})$
4:	Contract $X_{xa}^{\mu'} = C_{\lambda a}B_{x\lambda}^{\mu'}$	$\triangleright O(N_{occ}N_{bf}N_{aux}N_{CABS})$
5:	Contract $X_{\mu a}^{\mu'} = C_{\lambda a}B_{\mu\lambda}^{\mu'}$	$\triangleright O(N_{occ}N_{bf}^2N_{aux})$
6:	Contract $(\mu') = B_{\mu\nu}^{\mu'}D_{\mu\nu}$	$\triangleright O(N_{bf}^2N_{CABS})$
7:	Contract $(\nu') = [J^{-\frac{1}{2}}]_{\nu'\mu'}(\mu')$	$\triangleright O(N_{aux}^2)$
8:	Contract $K_{xy} = X_{xa}^{\mu'}X_{ya}^{\mu'}$	$\triangleright O(N_{occ}N_{aux}N_{CABS}^2)$
9:	Contract $J_{xy} = (xy\|\nu')(\nu')$	$\triangleright O(N_{aux}N_{CABS}^2)$
10:	Contract $K_{x\mu} = X_{xa}^{\mu'}X_{\mu a}^{\mu'}$	$\triangleright O(N_{occ}N_{bf}N_{aux}N_{CABS})$
11:	Contract $J_{\mu x} = B_{\mu x}^{\mu'}(\mu')$	$\triangleright O(N_{bf}N_{aux}N_{CABS})$
12:	Contract $K_{\mu\nu} = X_{\mu a}^{\mu'}X_{\nu a}^{\mu'}$	$\triangleright O(N_{occ}N_{bf}^2N_{aux})$
13:	Contract $J_{\mu\nu} = B_{\mu\nu}^{\mu'}(\mu')$	$\triangleright O(N_{bf}^2N_{aux})$

cleoside base-pairs. Comparing the times in Figure 4.1, we see that our ERI computation time is greater than that in that in the padded `DFHelper` code in all cases. This is due to the added number of memory addresses that needed to be populated in our implementation of the code. Choosing not to fill these addresses will necessitate a copy step which leads to problems for coulomb and exchange formation [10]. The ERI-wrapping code in `DFHelper` is highly optimized, and it fills a smaller amount of data. Our code includes the same optimizations but populates more data.

Besides the penalty in ERI tensor population, our formulation has decreased the amount of time spent in all operations as well as the total workflow time. This is due to the ability of our algorithm to avoid transformation against higher numbers of terms. The amount of time spent on metric contractions in our code is far more modest than the padded `DFHelper` implementation. The majority of savings in all cases in Figure 4.1 comes from this operation which scales as $O(N_{bf}N_{aux}^2N_{CABS})$ compared to the asymptotically smaller $O(N_{aux}N_{CABS}^2)$ time spent on ERI tensor population. Beyond those savings, as we can see in Figure 4.1 the time spent on constructing Coulomb and Exchange matrices is

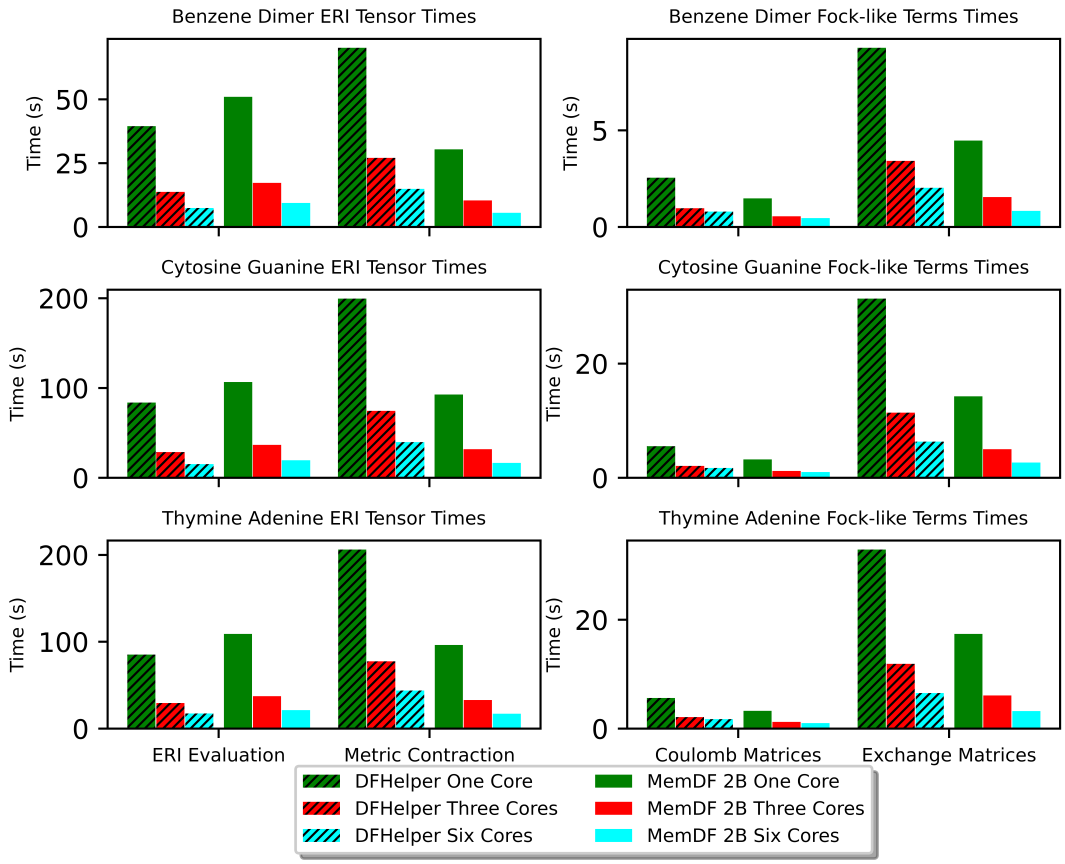

Figure 4.1: Timings of HF Calculations using `DFHelper` and Algorithm 10 Thymine-Adenine dimer in the aug-cc-pvdz basis and the appropriate ri bases using an Intel(R) Core(TM) i7-6800K CPU. ERI construction steps (left) are plotted separately because they take far longer. Timings were run on one, three, and six cores. Hashed times are the padded `DFHelper` implemenation, and the unhashed code is our implementation of Algorithm 10

halved.

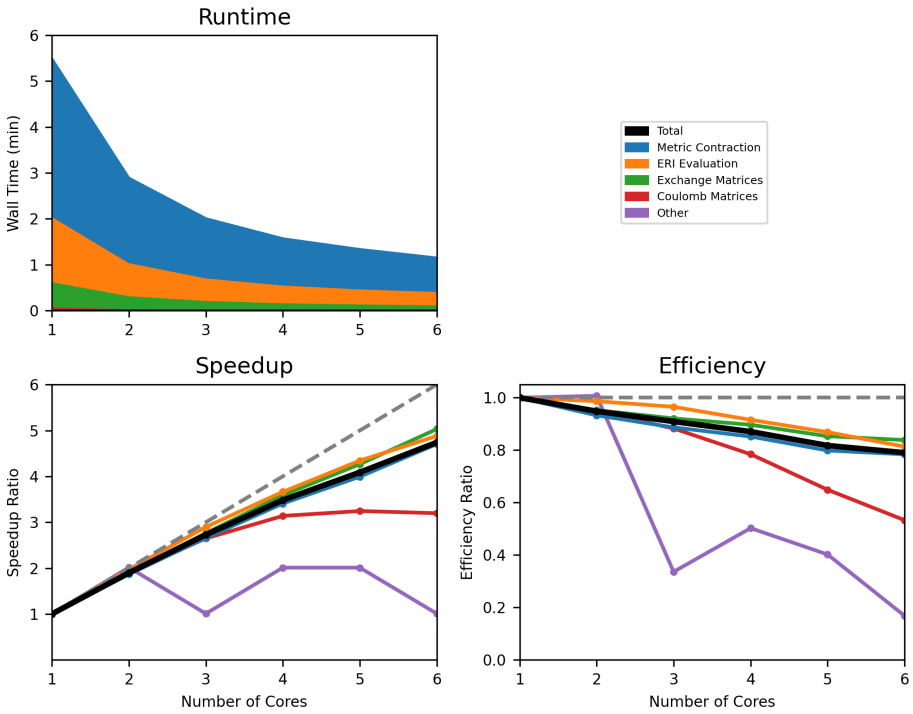

Figure 4.2: Timings of HF Calculations for a Thymine-Adenine dimer in the aug-cc-pvdz basis and the appropriate ri bases using padded `DFHelper` on a 40-ane with 3762 basis functions using an Intel(R) Core(TM) i7-6800K CPU.

Beyond comparing raw wall times, we must show that our algorithm does not sacrifice scaling as would concern us with knowledge of the increased amount of data involved. Timing and scaling information for the padded `DFHelper` workflow is in Figure 4.2. The scaling of the components of the calculations in this figure should match those of implementations of `DFHelper` in the previous Chapter 3.

These times are compared to the the timings for Algorithm 10 in Figure 4.3. We know from Figure 4.1 that we can expect lower times from these operations than from those in Figure 4.2.

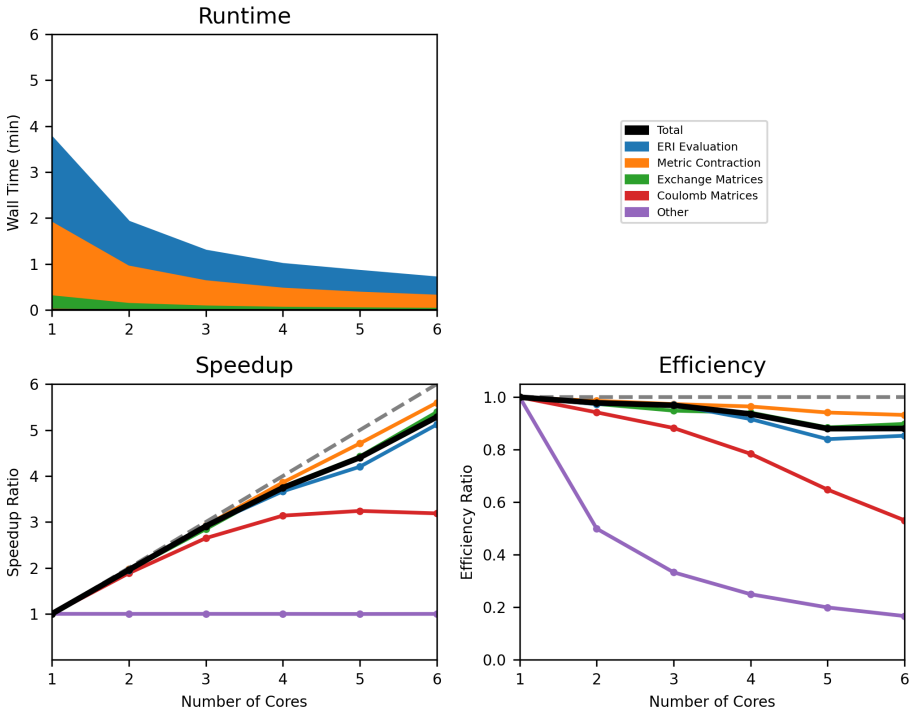

Figure 4.3: Timings of HF Calculations for a Thymine-Adenine dimer in the aug-cc-pvdz basis and the appropriate ri bases using padded `DFHelper` on a 40-ane with 3762 basis functions using an Intel(R) Core(TM) i7-6800K CPU.

Comparing the timings in Figure 4.3 to those in Figure 4.2, we can see that the parallelism the implementation of Algorithm 10 does not have worse scaling that the padded `DFHelper` workflow. Establishing this comparison is necessary for recommending this algorithm over the padded `DFHelper` workflow in general or for larger cases. Our code matches the padded `DFHelper` workflow in scaling and achieves lower runtimes.

CHAPTER 5
CONCLUSION

Formulations necessitating reduced computational effort and more parsimonious use of resources in general accelerate HF calculations and DFT calculations. We are also able to accelerate Fock-like calculations for explicitly correlated F12 methods. Our results for DirectDFJK show that it is necessary to make proper use of very high numbers of CPU cores when they are available while also formulating problems properly to avoid more computational effort than necessary with the limited resources available. Avoiding spurious copy steps and proper assessments of the working equations was used similarly to avoid such effort in implementing range-separated DFT in `DFHelper`. Reformulation and examination of working equations was also used to reduce the computational cost for SAPT-F12 terms. In all these efforts, it was necessary to probe the formulations of the working equations to achieve reductions in walltime and to avoid redundant work whenever possible. Careful consideration of the scaling of different parts of calculations in time and in memory within the scale relevant to realistic use cases was also necessary. In the future, other screening methods would aid in the methods used in Chapter 2, and, with more available memory, the calculations in chapter 5 could be made more generic.

Appendices

APPENDIX A

GTFOCK INTEGRATION

The GTFock project offers to accelerate Fock Matrix construction in HF calculations using distributed parallel architectures[26] and vectorized ERI calculation[27][28]. High performance open-source quantum chemistry is highly desirable to Psi4 and its users, so integrating the Fock matrix construction technology in GTFock is a goal for Psi4 developers. The GTFock package consists of four individual pieces: simint for ERI construction, libcint for managing ERI construction, GTMatrix for managing matrix operations on distributed architectures, and GTFock itself which uses the other packages for constructing Fock matrices and which has a short SCF program. A number of steps is necessary in this endeavor, namely installing GTFock on a high performance computing cluster and writing an interface between Psi4 and GTFock. Installing GTFock on Georgia Tech PACE and then Phoenix requires careful use of the proper Intel compilers and compiler options. On the Pace cluster, it was necessary to toggle between icc 2017 for simint and libcint and icc 2019 for GTMatrix and GTFock itself. It was also necessary to supply a modern gcc as well as gnu-prefix and sysroot options for all these compilations to proceed correctly. Once these steps are complete, it is necessary to ensure that code in Psi4 is able to link to GTFock binaries by modifying cmake inputs in Psi4 libraries. Because of the way that modules are constructed on the Phoenix cluster, a successful integration of the two packages will likely require inclusion of gnu-prefix and sysroot options as in Psi4 to avoid compile-time conflicts.

The programming of the interface consists of making molecular information in the basis set as well as information of the run-time architecture (i.e. the dimensions of the 2-D grid over which data is partitioned) available as inputs to GTFock. In the previous implementation of GTFock in Psi4, the interface was a C++ object that handled retrieving

the Coulomb and Exchange matrices as well as the Fock matrix, and the SCF density matrix from `GTFock`. A similar approach is recommended for the present to avoid compile-time conflicts. In `GTFock`, `MPI_Init` is called in the wrapping code to the Fock Matrix builder. The interface object will take the place of that code in the current integration, and Psi4 currently does not include calls to MPI Functions, so `MPI_Init` should be called in the `constructor` of that object, and `MPI_finalize` should be called in the `destructor` to minimize changes to `GTFock` and to avoid adding dependencies to Psi4. The Fock matrix construction code takes a `libcint` basis set `struct` and the `pfock struct` which holds information about the data of the partitioned Fock matrix held on a given node as input. The former can be constructed by providing basis set data, and the latter must be constructed with user input which can be handled as options in Psi4. In this case, it will be necessary to compile Psi4 using an `MPI` compiler so that it can be run using `mpirun`.

REFERENCES

[1] B. I. Dunlap, J. W. D. Connolly, and J. R. Sabin, "On the applicability of lcao-xα methods to molecules containing transition metal atoms: The nickel atom and nickel hydride", *International Journal of Quantum Chemistry*, vol. 12, no. S11, pp. 81–87, 1977. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qua.560120813.

[2] B. I. Dunlap, J. W. D. Connolly, and J. R. Sabin, "On some approximations in applications of xα theory", *The Journal of Chemical Physics*, vol. 71, no. 8, pp. 3396–3402, 1979. eprint: https://doi.org/10.1063/1.438728.

[3] J. L. Whitten, "Coulombic potential energy integrals and approximations", *The Journal of Chemical Physics*, vol. 58, no. 10, pp. 4496–4501, 1973. eprint: https://doi.org/10.1063/1.1679012.

[4] H.-J. Werner, F. R. Manby, and P. J. Knowles, "Fast linear scaling second-order møller-plesset perturbation theory (mp2) using local and density fitting approximations", *The Journal of Chemical Physics*, vol. 118, no. 18, pp. 8149–8160, 2003. eprint: https://doi.org/10.1063/1.1564816.

[5] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*, First. Mineola: Dover Publications, Inc., 1996.

[6] J. P. Perdew, M. Ernzerhof, and K. Burke, "Rationale for mixing exact exchange with density functional approximations", *The Journal of Chemical Physics*, vol. 105, no. 22, pp. 9982–9985, 1996. eprint: https://doi.org/10.1063/1.472933.

[7] W. Kohn and L. J. Sham, "Self-consistent equations including exchange and correlation effects", *Phys. Rev.*, vol. 140, A1133–A1138, 4A Nov. 1965.

[8] M. Head-Gordon, J. A. Pople, and M. J. Frisch, "Mp2 energy evaluation by direct methods", *Chemical Physics Letters*, vol. 153, no. 6, pp. 503–506, 1988.

[9] D. G. A. Smith, L. A. Burns, A. C. Simmonett, R. M. Parrish, M. C. Schieber, R. Galvelis, P. Kraus, H. Kruse, R. Di Remigio, A. Alenaizan, A. M. James, S. Lehtola, J. P. Misiewicz, M. Scheurer, R. A. Shaw, J. B. Schriber, Y. Xie, Z. L. Glick, D. A. Sirianni, J. S. O'Brien, J. M. Waldrop, A. Kumar, E. G. Hohenstein, B. P. Pritchard, B. R. Brooks, H. F. Schaefer, A. Y. Sokolov, K. Patkowski, A. E. DePrince, U. Bozkaya, R. A. King, F. A. Evangelista, J. M. Turney, T. D. Crawford, and C. D. Sherrill, "Psi4 1.4: Open-source software for high-throughput quantum chemistry", *The Journal of Chemical Physics*, vol. 152, no. 18, p. 184 108, 2020. eprint: https://doi.org/10.1063/5.0006002.

[10] M. Scheiber, "Optimizing computational kernels in quantum chemistry", Master's thesis, Georgia Institute of Technology, 2018.

[11] F. Weigend, "A fully direct ri-hf algorithm: Implementation, optimised auxiliary basis sets, demonstration of accuracy and efficiency", *Phys. Chem. Chem. Phys.*, vol. 4, pp. 4285–4291, 18 2002.

[12] R. A. Kendall, T. H. Dunning, and R. J. Harrison, "Electron affinities of the first-row atoms revisited. systematic basis sets and wave functions", *The Journal of Chemical Physics*, vol. 96, no. 9, pp. 6796–6806, 1992. eprint: https://doi.org/10.1063/1.462569.

[13] A. Savin and H.-J. Flad, "Density functionals for the yukawa electron-electron interaction", *International Journal of Quantum Chemistry*, vol. 56, no. 4, pp. 327–332, 1995. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qua.560560417.

[14] G. D. Scholes, G. R. Fleming, A. Olaya-Castro, and R. Van Grondelle, "Lessons from nature about solar light harvesting", *Nature chemistry*, vol. 3, no. 10, p. 763, 2011, doi:10.1038/nchem.1145.

[15] K. Mazmanian, K. Sargsyan, C. Grauffel, T. Dudev, and C. Lim, "Preferred hydrogen-bonding partners of cysteine: Implications for regulating cys functions", *The Journal of Physical Chemistry B*, vol. 120, no. 39, pp. 10 288–10 296, 2016, PMID: 27635780. eprint: https://doi.org/10.1021/acs.jpcb.6b08109.

[16] T. Yanai, D. P. Tew, and N. C. Handy, "A new hybrid exchange–correlation functional using the coulomb-attenuating method (cam-b3lyp)", *Chemical Physics Letters*, vol. 393, no. 1, pp. 51–57, 2004.

[17] J.-D. Chai and M. Head-Gordon, "Systematic optimization of long-range corrected hybrid density functionals", *The Journal of Chemical Physics*, vol. 128, no. 8, p. 084 106, 2008. eprint: https://doi.org/10.1063/1.2834918.

[18] T. Shiozaki, M. Kamiya, S. Hirata, and E. F. Valeev, *J. Chem. Phys.*, vol. 129, p. 071 101, 2008.

[19] J. Noga, S. Kedzuch, J. Simunek, and S. Ten-no, *J. Chem. Phys.*, vol. 128, p. 174 103, 2008.

[20] W. Kutzelnigg and J. D. Morgan, *J. Chem. Phys.*, vol. 96, p. 4484, 1992.

[21] L. Kong, F. A. Bischoff, and E. F. Valeev, "Explicitly correlated r12/f12 methods for electronic structure", *Chemical Reviews*, vol. 112, no. 1, pp. 75–107, 2012, PMID: 22176553. eprint: https://doi.org/10.1021/cr200204r.

[22] K. Szalewicz, "Symmetry-adapted perturbation theory of intermolecular forces", *WIREs Computational Molecular Science*, vol. 2, no. 2, pp. 254–272, 2012. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcms.86.

[23] H.-J. Werner, T. B. Adler, and F. R. Manby, "General orbital invariant mp2-f12 theory", *The Journal of Chemical Physics*, vol. 126, no. 16, p. 164 102, 2007. eprint: https://doi.org/10.1063/1.2712434.

[24] H. E. A., *Adv. Quantum Chem.*, vol. 1, p. 1, 1964.

[25] E. F. Valeev, "Improving on the resolution of the identity in linear r12 ab initio theories", *Chemical Physics Letters*, vol. 395, no. 4, pp. 190–195, 2004.

[26] H. Huang, C. D. Sherrill, and E. Chow, "Techniques for high-performance construction of fock matrices", *The Journal of Chemical Physics*, vol. 152, no. 2, p. 024 122, 2020. eprint: https://doi.org/10.1063/1.5129452.

[27] B. P. Pritchard and E. Chow, "Horizontal vectorization of electron repulsion integrals", *Journal of Computational Chemistry*, vol. 37, no. 28, pp. 2537–2546, 2016. eprint: https://www.onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.24483.

[28] H. Huang and E. Chow, "Accelerating quantum chemistry with vectorized and batched integrals", in *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC18)*, Nov. 2018, pp. 41–50.